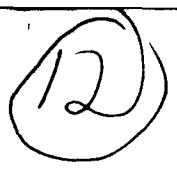
AD-A271 646



DTIC
ELECTE
OCT 2 9 1993
S A D

# Decoupled Computer Architectures for Very High-Speed Technologies

Co-P.I.s: Steven E. Butner and Stephen I. Long

**FINAL REPORT**
**DARPA, ARPA Order # 6356**
**July 1, 1988-June 30, 1992**

**ECE Technical Report #93-19**

This document has been approved for public release and sale; its distribution is unlimited.

**Department of Electrical and Computer Engineering**
**University of California, Santa Barbara, CA 93106-9560**

**October 1993**

93-24555

93 10 15 103

# Decoupled Computer Architectures for Very High-Speed Technologies

**Final Report**

**Department of Electrical and Computer Engineering**
**University of California**
**Santa Barbara, Calif. 93106**

**Co-Principal Investigators:** **Steven E. Butner — butner@ece.ucsb.edu**
**Stephen I. Long — long@ece.ucsb.edu**

**Faculty Investigators:** **Nadir Dagli**
**John P. J. Kelly**
**George Matthaei**

**Performance Period:** **July 1, 1988 - June 30, 1992**

**Sponsored by the:** **Defense Advanced Research Projects Agency**
**ARPA Order Number 6356**

**Monitored by:** **Office of Naval Research**

Accesion For
NTIS
DTIC
Unannounced
Justification
By
Distribution /
Availability Codes
Dist | Avail and / or Special
A-1

Contract Number: N00014-88-K-0497

# 1.0 Overview and Summary

This report is a summary of the research activities for the project "Decoupled Computer Architectures for Very High-Speed Technologies" which has been undertaken from July 1, 1988 through July 31, 1992 under Defense Advanced Research Projects Agency (DARPA) CSTO sponsorship..

## 1.1 Technical Objectives of the Project

The long-term goal of this project is to develop the capability to *apply* and *effectively use* emerging high-speed IC technologies such as gallium-arsenide. Success in this endeavor requires experience with circuits, packaging, and architecture, not only in the *new* technologies, but also in current MOS IC technologies.

The fundamental problem addressed here is that though GaAs systems can now (or soon) quite readily be built with 100-150 MHz clock rates, the scaling *beyond* that speed is greatly complicated due to two effects:

- some subsystems within a computer can support faster clock rates without much special consideration, but there are subsystems (most notably *memory* and *communications*) that cannot readily be made to follow the faster clock rates.
- conventional computer architecture is not capable of effectively dealing with high latency operations. As clock rates for some (but not *all*) of the subsystems of a computer increase, the relative latency of the remaining subsystems grows.

## 1.2 Main Architectural Work and Roadmap

We have left the 100-150 MHz GaAs processor work for others, seeing it as evolutionary (not necessarily *easy*, but sure to happen), choosing instead to work on the fundamental problems above.

It was our belief when our program started that rather significant architectural changes were necessary in order to address these problems. We still believe this to be so. Following is a summary of the research ideas we have pursued in the computer architecture portion of this contract.

### 1.2.1 Multi-phase instruction processing

Break the instruction processing job into a *transaction style* so that servers (processing pipelines) can pull work from queues. This has several major effects:

- we can choose the number of servers for each type of transaction in order to closely match the average flow rate into each queue. If the queues are implemented as

FIFOs and sized correctly, then complex flows with irregular bursts are converted into predictable average flows that can feed processing pipelines.

- the use of FIFOs provides an opportunity to break the traditional hard synchronization of global clocks and use different local clocks for each subsystem.
- we have the opportunity for tolerating failures since the servers are all identical and, hence, can stand in for one another.
- both superscalar and superpipelining concepts can be integrated under this decoupled, transaction-oriented philosophy (by changing the number of units that service each type of queue and the level of concurrency of processing applied to a given instruction stream).

#### 1.2.2 Determine and develop an appropriate interconnection network for this type of machine

This has resulted in the creation, modeling/analysis, design, and fabrication and demonstration of the SHUNT network, a circuit-switched, highly-reliable technology-scalable tree-like ICN. A 16-node cluster of the SHUNT network has been prototyped.

#### 1.2.3 Deal with dependencies within the instruction stream

This is the classical problem that causes pipelines to have bubbles. The difference in our case is that the machine we are creating can handle latency ... so we have an additional free variable to use in solving the dependency problem.

The initial approach was to interleave instruction streams so that every transaction in any given FIFO is part of a different independent process (hence cannot have a dependency). This solves the dependency problem, but requires a large number of concurrent processes. This makes the machine better for some types of problems than for others. The required number of processes and their characteristics is remaining research. The optimum point for the number of concurrent processes is a function of instruction set, compiler technology, memory size and interleave, and is complex to find.

There are many design knobs to turn in such a system. The ALEWIFE/SPARCLE project at MIT has studied a latency-absorbing approach, but theirs is quite different from ours --- focussing primarily on communication latency. Our project has attempted to overlap *all types* of latency.

#### 1.2.4 Further leverage the architectural improvements by high-performance circuit techniques

The GaAs circuit work in this project has always had a consistent focus: reduce power dissipation without sacrifice in density or manufacturability. The two-phase dynamic FET logic family (TDFL) developed under this project meets these objectives while operating at clock rates of 1 GHz. Power dissipation 20 times less(!) than static CMOS has been demonstrated in MSI-level test circuits. The self-latching character of TDFL allows highly-efficient implementation of pipelined circuits and its compatibility with low-power static GaAs DCFL is also a bonus. We want to apply TDFL to critical functions needed in our decoupled computing system to increase throughput (e.g. FIFOs, arithmetic units, and register files).

We will continue to investigate innovative circuit techniques that can get a high percentage of the performance possible out of GaAs. We want to run clocks as fast as 1 GHz if possible. As low power GaAs VLSI now seems feasible, higher-level design tools will be employed in the design of large GaAs chips (following the approach at Michigan).

#### 1.2.5 Given a decoupled style, develop the most appropriate instruction set for that type of machine.

We've done less work on this than on the other parts, simply due to time and budget limitations. Instruction set studies require vast amounts of computer time. It is our intention to use the prototype DART processor as a vehicle to support this activity. The prototype has a programmable instruction set, as well as adjustable memory interleaving, etc. for this very purpose.

#### 1.2.6 The DART Prototype

As a driver for our research we have undertaken the project of creating, developing, refining, and implementing a proof-of-concept prototype of a scalable multiprocessor architecture that is well-suited to implementation in gallium-arsenide or other very high-speed technologies. Primary subgoals of this effort were to:

- Investigate the use of *decoupling* to solve the difficult problems related to the use of very high-speed circuitry. A distributed, decoupled computer philosophy and organization has emerged that provides the basis for a parallel system with the properties of scalability, ultra-high utilization, locally synchronous - globally asynchronous timing, single clock inter-process context switching, and architectural support for memory hot spots and load balancing.
- Investigate, develop, and refine novel dynamic GaAs circuits that are well-suited to implement deeply-pipelined computer processors such as are used in decoupled machines.
- Develop and verify models and efficient approximation techniques for analysis of crosstalk and wave effects in very high-speed digital ICs.
- Experimentally verify the architectural research ideas via construction of a proof-of-concept prototype.

### 1.3 Technology Transfer

Because the architectural work on this project has been quite fundamental and because it is not yet ready for product form, our main technology transfer path has been in the training of PhD students. Since program start, the architecture group alone has graduated 6 PhDs. Four are professors in computer engineering programs (Cal Poly, Cal State Fresno, Texas A&M and Naval Postgraduate School), , and two have formed a private computer company. The other groups within our project have also generated their share of PhDs. A circuit's group PhD is now at the Rockwell Science Center. Of course, there are numerous MS-level graduates as well, feeding such companies as National, AMD, HP, DEC, Sun, Amdahl, Intel, ...

We have interacted with several other DARPA groups over the last several years, specifically: Utah, MOSIS, ISI/APT(Parker), JPL, Mayo(Gilbert) RPI, Caltech, and Johns

Hopkins.

Perhaps the most visible form of technology transfer that has emerged from DARPA-sponsored research at UCSB is the textbook "Gallium-Arsenide Digital Circuit Design" (co-authored by Long and Butner, McGraw-Hill,-1990), the Magic technology files and cells for GaAs, the UCSB high-speed test board (we've delivered over 25, mainly to DARPA contractors), and the various short courses we've held to help the community get up to speed on GaAs.

## 1.4 Research Highlights

The project has been organized as three cooperating subgroups: architecture, circuits, and wave effects. The following are research result highlights.

- One 16-processor cluster of the SHUNT interconnection network is up and running tests and diagnostics.
- The kernel processor board — housing the I/O services and cluster-local support for DART's distributed operating system — is fully operational at this time. A primitive operating system is implemented and running.
- The DART processor board design is complete and 8 processor cards are implemented and are operational. We have had a great deal of trouble with connectors and connection reliability at system level, however, due to problems with non-delivery by vendors and the eventual decision to use a complex hand-made connector assembly. This hand-made assembly was made by Jeff Lacoss' group at ISI. It is unfortunate that the original connector vendor never delivered the critical component that was needed. Once boards had been made and 6-months of valuable integration & test time had been lost in waiting for delivery, the decision was made to build our own connector. This was a lot of work and we appreciate the efforts of the ISI group. But, the reliability of the interconnections has not been high. We continue to observe significant failures in the backbone ICN-to-processing node connections.
- Our PC-based program to compute lossy transmission line parameters and the time response of high-speed digital circuit interconnects has been improved and completed. Since the last reporting period, approximate models for the resistance variation of long on-chip interconnects as a function of frequency have been developed. The models have been compared with published analysis and with measured data. The time response program can now accurately deal with lossy interconnects. Sample copies of the program have been sent to other institutions for evaluation.
- Design rules have been formulated for lossy on-chip interconnections through the application of our accurate equivalent circuit model for coplanar lines. HGaAs-3 line parameters and signal propagation were calculated. Signal integrity has been degraded with the new process, and new design rules were formulated.
- A fully dynamic, ultra low-power, high speed circuit family for GaAs MESFET technologies has been developed. A fully dynamic four-bit ripple carry adder was evaluated as one of several demonstration circuits. Operation at 500 MHz required only 1 mW of power.
- The UCSB circuit simulator which is based on new forward integration techniques is producing solutions to ever larger networks, now including MOSFETs. Device mod-

els have been ported from SPICE3 so that our new simulator is built on tried and true, well understood primitives.

# 2.0 Architecture Group — Projects and Results

## 2.1 The DART experimental prototype

Steven Butner, Steve Jordan, Joy Shetler, Juraj Malek, Jim Dodd, Wayne Yamamoto
Computer Architecture and Test Laboratory
University of California, Santa Barbara

### 2.1.1 Goals

Among the primary goals of this research program was the study of the use of decoupling in high-performance processors. In support of that goal we undertook the construction of an experimental multicomputer — DART. Rather than pushing the limits on a particular decoupled processor design, we decided to implement a flexible vehicle for

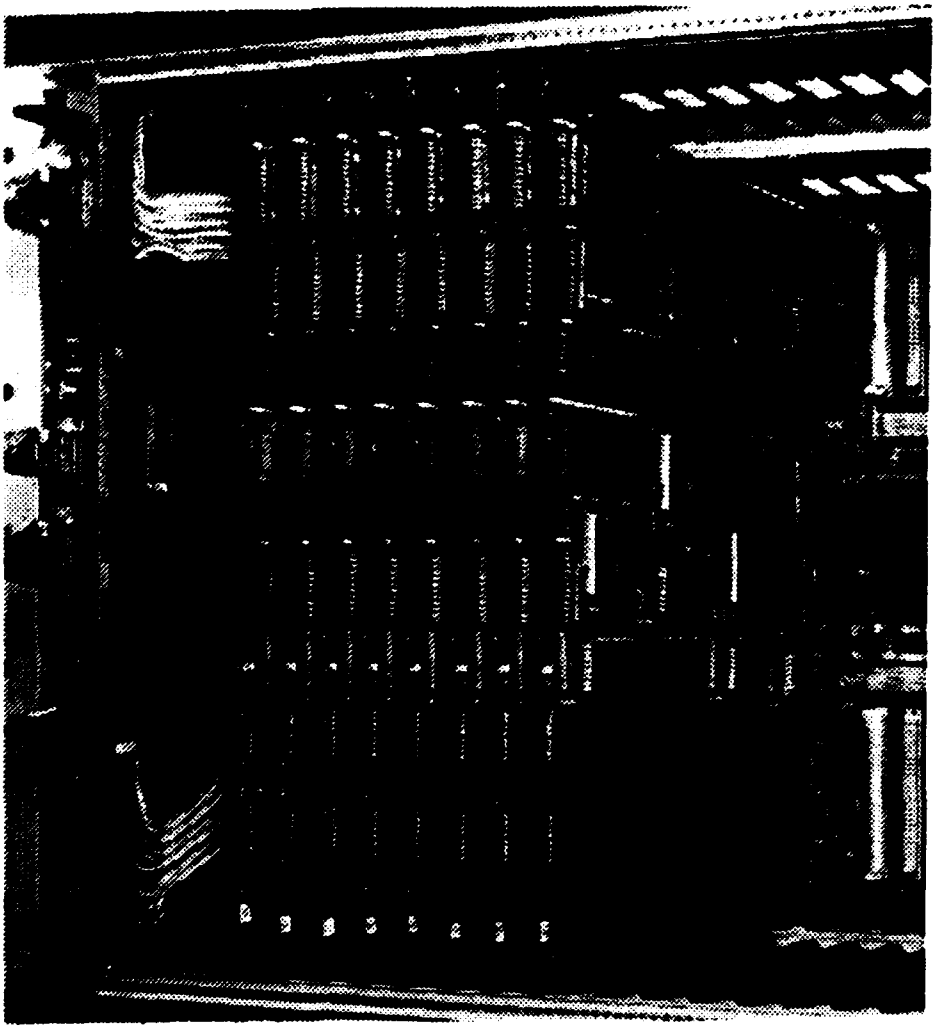


PHOTO 1. A 16-node Cluster of the DART Prototype with eight DART processors installed.

experimentation, one that will provide the opportunity to study the interplay of many design parameters. This machine has a highly programmable internal behavior so that we can investigate the relative payoffs and costs of various parameters relating to decoupling, e.g. memory bank interleaving, process migration policies/techniques, interconnection network issues such as packet size and degree of buffering, even instruction set design.

Throughout the program we have focussed significant energies on the design and construction of our prototype DART processor as well as its novel interconnection network, SHUNT. In the short sections that follow, we briefly discuss the individual components of DART.

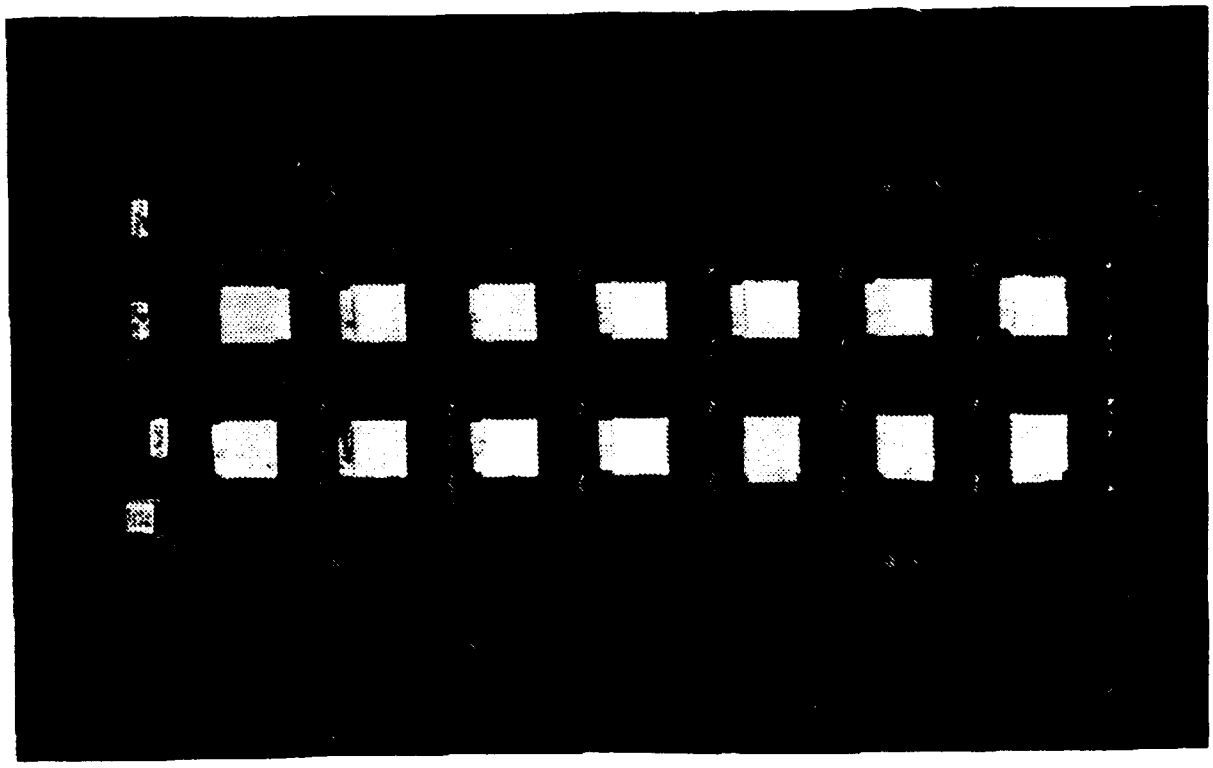


**PHOTO 2.** View of SHUNT ICN switching card containing 27 full-custom SCMOS chips (double-sided; back side not shown)

**SHUNT interconnection network:** This new interconnection network is the backbone of the multicomputer. SHUNT is a tree-structured, scalable, circuit-switched network. It is built in 16-node clusters (depicted in photo 1), with each level of the tree fanning out 16 ways. In addition to a strict tree-structure, SHUNT also provides left- and right-neighbor connections that form a ring across each level of the tree. Associated with each cluster is a dedicated operating system kernel processor. This processor provides I/O and cluster-local control services within the context of a larger distributed system. We

acknowledge the support of ISI's Advanced Packaging group (Bob Parker Jeff Lacoss, Diane Delute, John Granacki) for their help in the packaging of SHUNT and the circuit board routing of the DART processors. The SHUNT packaging was a challenge, like most interconnection networks, due to high pin counts and small chip sizes.

The SHUNT ICN is made from three custom chip types: the SHUNT controller, the bit-sliced crossbar, and the BRIDGE ICN-to-processor interface.

- Crossbar — this is the simplest of the SHUNT chips. Each crossbar chip implements a 23-by-23-by-1 crossbar switch. In order to implement an $n$-bit data path, $n$ such chips would be used. In our implementation, we have chosen to create a 16-bit data-path with 6 bits of modified Hamming code. Together with 4 bits of handshake and control, our design requires 26 crossbar chips for each 16-node switching cluster. The crossbar chip is implemented in 1.2 micron scalable CMOS and is packaged in an 84-pin J-leaded ceramic chip carrier. Unfortunately, when our chips were packaged by Halcyon, the packager used the wrong type of vise to hold the packages and severely crushed the J-leads, so much so that they would no longer correctly mate with the sockets. Since they packaged *all* of our die in this way, we had no choice but to try to repair the damage by hand. The result is workable, but the connections are not solid and we often find that after tracing a problem for several hours, it is just a package pin that is not making contact with its socket.
- SHUNT controller — the "brains" of each switching cluster. This chip makes all of the decisions regarding when to cut through new paths and which ones to do before others, etc. The controller is implemented in 2.0 micron SCMOS and is packaged in the same way as the crossbars (including the unfortunate crushed J-lead scenario). There is only one of these chips per cluster, however, so the crushed lead problem is not as serious as it is with the crossbars.
- BRIDGE interface — this chip serves as a convenience, presenting the appearance to a processor of a 32-bit bus when in fact the ICN is only 16-bits wide (plus 6 bits of modified Hamming code for error detection and correction). The BRIDGE is fully asynchronous, talking to the SHUNT ICN on one side and to a processor on the other. We still have problems with this chip at this time. The first fab of it was mostly functional, but there were some problems relating to its asynchronous nature which prevented us from using it without changes. The second fabrication, which should have fixed the asynchronous problems, had serious latchup problems. We traced the latchup problems to a set of buffers that were added at the last minute (without simulating!) that had their plugs connected backwards. Plugging problems are not easy to discover; our CAD tools fall short in this regard. We have implemented a small printed circuit daughter card to provide most of the functionality of BRIDGE and to allow the DART system to work around the non-functional BRIDGE chip.

### 2.1.2 DART Implementation and Results

At the present time, we have 8 DART processor cards installed in a 16-node SHUNT ICN cluster. The cluster also has a kernel processor and a SUN-3/110 host for down-loading and I/O. The DART cards each have 4 microprocessors and 4 Mbytes of DRAM on them; this structure provides us with the opportunity to emulate a wide variety of decoupled structures. The DART cards and kernel processor are running at 32 MHz and we can communicate between them via the SHUNT ICN.

Our detailed simulations of various decoupled designs have shown that, as hoped, the idea of decoupling has merit in high clock rate systems. There are many closely interacting factors, however, that influence design choices both in degree and in the best location for application. It is our hope to use the programmability of the DART to experimentally determine optimum choices in the design of decoupled processors.

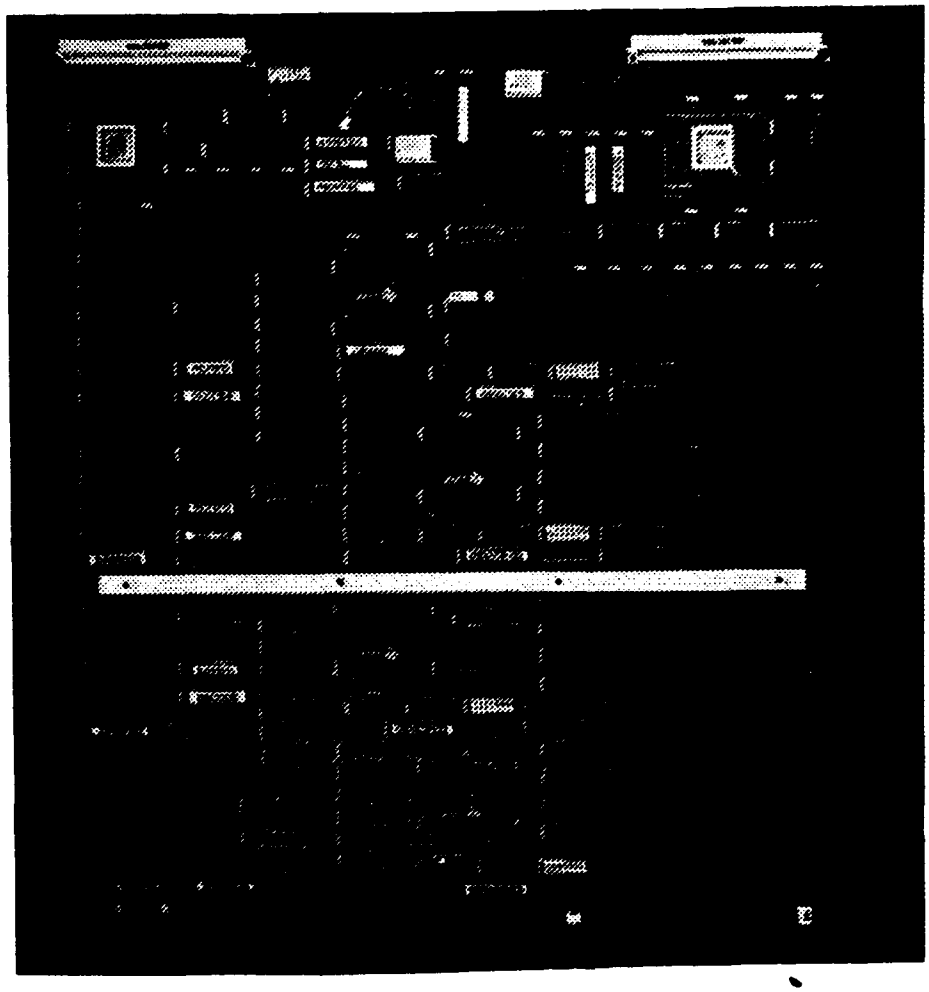


PHOTO 3.

DART processor card containing 4 programmable subsystem processors (each with a 1 Mb portion of a shared global memory space), inbound and outbound SHUNT-ICN ports with BRIDGE interfaces, on-card packet queues and intra-subsystem ICN, and FIFO decoupling.

## 2.2 SHUNT Interconnection network board — JTAG Implementation

Juraj Malek, Steven Butner, Steve Jordan, Jim Dodd
Computer Architecture and Test Laboratory
University of California, Santa Barbara

### 2.2.1 Introduction.

The JTAG (Joint Test Action Group) standard is IEEE system-level standard 1149.1, designed for chip and on-board connectivity testing purposes. Its main idea is to place a scan shift-register stage at the chip boundaries to improve controllability and observability. The shift register stages are placed adjacent to every input and output pin of a chip. These cells around the periphery of the IC are connected to form a shift-register path called the Boundary Scan Register (BSR). During normal operation data can flow directly in/out of the chip through the boundary scan cells. During testing (while remaining fully interconnected on the system board):

- boundary scan cells at output pins can be used to drive and desired signal values to the outside environment of the chip
- cells at input pins can drive chosen test stimuli into the internal logic of the chip
- signals flowing in and out of the chip during normal operation can be synchronously captured and the sample can be examined.

### 2.2.2 Brief JTAG description

The implementation of the JTAG test logic includes three key components:

- Test Access Port (TAP) controller: This block responds to the control signals applied at the test pins of a chip and generates the clocks and control signals for internal test circuits.
- Instruction Register: A two (or more) bit register that is serially loaded with an instruction that selects the JTAG operation to be performed.
- Test Data Register(s): This is a set of shift-registers. The data (test vectors) can be shifted in or the contents of any data register can be shifted out of the chip.

### 2.2.3 Our JTAG implementation

We have implemented the JTAG standard in the design and fabrication of two types of chips — the SHUNT controller and crossbar, which are two of the three custom chip types making up the SHUNT interconnection network.

There were three main reasons for the implementation:

- There are not enough pins on the controller chip to load in the position vector which is necessary as part of its initialization. JTAG was used to provide this data loading capability.
- The surface-mount technology, double sided board, and the location of some chips on the SHUNT interconnection network boards made it impossible to place measuring probes on the pins of the chips.

- There are simply too many pins and signals on the board to monitor by physically connecting probes (nearly 1600 signal pins are in the JTAG boundary scan path).

The heart of the SHUNT network is the controller chip. It is responsible for the arbitration of connection requests, scheduling of interconnections and configuration of the crossbar switches, so that a complete path can be made from input port to the requested output port of the network. The crossbar chips implement an array of 23x23 switches capable of interconnecting any permutation of inputs to outputs.

The crossbar's implementation of the JTAG standard involves the basic mandatory features:

- Instructions: BYPASS, EXTEST, SAMPLE/PRELOAD.
- Test data registers: bypass register and boundary scan register (59 bits long).

Because of the necessity to load the position vector into the controller chip via JTAG, the position vector register and the INTEST instruction were added to the design of the JTAG part of the controller. The boundary scan register of the controller is 65 bits long.

A 15 bit long position vector determines the position of a particular controller within the scalable tree-structured network. The position vector must be loaded via JTAG before system operation begins.

Both types of chips include five JTAG test pins: TRST, TMS, TCLK, TDI, TDO.

There is one controller chip and 26 crossbars on the interconnection board. Thus the complete boundary scan path is 65 + 26x59 = 1599 bit long.

### 2.2.4 The JTAG control program.

To be able to load the position vector register, drive input/output pins of the chips, capture test data, as well as monitor and examine live signals, we have created a program to manage all JTAG operations. The programs allows the user to:

- reset the controller and all crossbar chips
- test both bypass and complete scanpath-chain connectivity of all the chips and perform a test of the position vector register
- load the operational position vector into the controller
- remap the interconnection network, specifying which ports are active and which are out of service
- drive user-specified groups of input and output pins of the chips. The user can interactively select the chips, signals, and values to drive.
- capture and display the values of all the signals of all the chips for analysis and debugging.

We have designed the JTAG control program to be as general as possible. It has a modular structure, consisting of independent functions and data structures which can be used for any other chip configuration adhering to the JTAG standard. To port to a new config-

uration, only the definition part of the program containing the JTAG specific constants and parameters of the chips (scanpath length, instruction codes, etc.) and the information about the order of the chips in the scan chain needs to be changed.

### 2.2.5 Looking back on our JTAG implementation

The IEEE Standard 1149.1 (JTAG) was primarily developed for chip testing and on-board connectivity testing throughout the manufacturing process. Unlike its original intent, we have been using the JTAG interface capabilities mainly for loading the position vector register and for monitoring the signals for analysis and debugging purposes.

We present here some differences of our JTAG implementation from the standard, the bugs we discovered in the design, and further suggestions:

- During the EXTEST instruction in our implementation, both input and output pins of the chips are driven by the data in the boundary scan register. The standard specifies that only output pins should be driven. In our implementation, the signals are (incorrectly) applied to the internal system logic causing it to react and drive the output pins accordingly after leaving the EXTEST instruction. This behavior may have undesirable effect on other parts of the system.
- We call ... struction during which the position vector register is loaded, the INTEST (internal test) instruction. According to the standard, the test of internal system logic of the chip should be performed during this instruction. So the name INTEST in our terminology may be misleading.
- During activation of the INTEST instruction, when the position vector is being loaded, the data in the boundary scan register are driven into and out of the controller chip (as in the EXTEST instruction). This is a bug in our JTAG controller design, where the "Mode" control signal is decoded incorrectly. This signal should be active only when the EXTEST instruction is present.
- Like all other input pins, the controller chip's system *reset* input is located at the boundary scan path. This is quite normal. We made the mistake, however, of connecting the JTAG reset signal (TRST) to the chip reset. With this error, whenever in the EXTEST or INTEST instruction the value 0 (active value of the chip Reset signal) is loaded into the BSR at the position of the Reset pin, the controller and the JTAG test logic gets reset as well. This is a serious error, yet one that is quite natural to make. We have been able to work around the problem in our particular case (by keeping track of the values in the boundary scan register).
- Perhaps the most serious bug in our JTAG implementation concerns a misunderstanding of how incoming data is routed. The description of the standard is not crystal clear about this point. In our implementation, all incoming data (TDI) — no matter which register it is destined for — always gets shifted into the boundary scan register. JTAG is quite flexible and even this bug can be worked around. But, it would help if only the test data register corresponding to the current instruction would be allowed to change contents. We will fix this in future JTAG implementations.

### 2.2.6 Conclusion

The implementation of the JTAG standard in our chip design made a major contribution

towards the testability and performance of the designed chips and the board as a whole. Without the JTAG implementation it would have been physically impossible to test the internal system logic of the chips, to debug the board and its software control, and to monitor the large number of signals on the interconnection network's double sided SMT printed circuit board.

## 2.3 SPArTAn: A Scientific Processor Architecture for Task-Partitioned Applications

James Dodd, Steven Butner
Computer Architecture and Test Laboratory
University of California, Santa Barbara

### 2.3.1 Introduction

A study of current research projects throughout the spectrum of scientific study indicates a rapidly growing reliance on computers to verify and extrapolate existing knowledge. The relatively limited power of current computer technology is holding back scientific progress in many fields. These limits are directly related to the hardware architecture and software environment of existing scientific computing technology.

Many current processor architectures exhibit a high level of inefficiency in pipeline usage and correspondingly high average CPI's (clocks per instruction). This translates directly to performance which is much lower than the theoretical maximum. For example, the SPARC2 processor theoretically has a CPI of 1.0. However, the CPI recorded for typical applications is well above 2.0. These inefficiencies are due to branch delays, pipeline dependencies, and memory delays.

This research first concentrated on analyzing a number of scientific applications running on typical RISC workstations. Analysis of the dynamic instruction mix indicated a number of important characteristics of scientific applications which were not expected.

- The percentage of floating-point instructions was much lower than expected. Even though the majority of variables declared in a program are floating-point, the percentage of instructions to operate on these data types was below 5%. This was seen in all of the applications studied. The majority of operations were integer and load/store. This is caused by the large amount of looping and data movement needed. In addition, a large amount of branching instructions were found. These branches also caused a large number of empty cycles while the branch target was calculated.
- Another important characteristic of the applications studied was the ease with which the model could be partitioned into a number of independent tasks. This characteristic suggests that some amount of parallelism or multi-tasking will improve performance.
- Memory usage analysis of scientific applications indicated that the amount of data memory was several orders of magnitude larger than the amount of instruction memory needed.

### 2.3.2 The SPArTAn Architecture

From this study, SPArTAn (Scientific Processor Architecture for Task-Partitioned Applications) evolved. This processor has a number of basic features which are intended to improve the execution of scientific applications. SPArTAn does not define a specific implementation. Instead, it offers an architectural paradigm for improving both execution and programming efficiency. The ideas here are readily implementable; they can be made to work in any of several technologies (SCMOS as well as GaAs) in order to provide a significant improvement over existing RISC processors.

SPArTAn is back-end processor, intended to be operated from a commercially available workstation such as a SPARC or RS/6000. This organization allows for an optimal cost/performance ratio. The front-end workstation provides a sophisticated operating system and user interface, as well as a large file system. SPArTAn provides a maximum performance using standard technology. In addition, SPArTAn is a heavily pipelined RISC processor.

One of the most important design issues with any pipelined processor is the reduction in pipeline dependency penalties. The existence of these dependencies can cause a significant reduction in pipeline utilization and overall performance. These dependencies can be of the form of data references or flow-control. Current processors have a number of complicated techniques to minimize pipeline dependencies or eliminate them (e.g. hardware scoreboarding, pipeline interlocking, and instruction scheduling). These techniques are typically only partially successful. For example, branch-delay slot filling typically only fills between 50% and 60% of all branch delay slots with useful instructions. This can mean a 10 to 15% reduction in overall efficiencies. Here at UC Santa Barbara, the DART processor uses Interleaved-Instruction Streams (IIS) to eliminate pipeline dependencies completely. By inserting only one instruction per task into the pipeline there can be no pipeline dependencies. The throughput of DART is a function of how many tasks are running. This approach is simple and easily implemented. The only drawback to this approach is the need for a large number of tasks in order to fill the pipeline.

A logical extension of IIS has been used in SPArTAn. This technique is called Instruction-Block Interleaving (IBI). Instead of inserting only one instruction into the pipeline per task, a task inserts the maximum number of instructions possible without causing a pipeline dependency. This minimizes the number of tasks required for a given pipeline length, while still maintaining an optimal pipeline usage. If done correctly, a small number of tasks can provide 100% utilization of the pipeline with no empty cycles. In other words, a CPI of 1.0 is attainable with a single execution unit.

The drawback to this approach lies in developing a mechanism to allow for detection of impending pipeline dependencies before the offending instruction is inserted into the pipeline. This mechanism has been developed through a tight coupling of software and hardware in SPArTAn. While one task is running, another task must be waiting in the control unit to begin execution. When the running task encounters an impending pipeline dependency it stops execution. The waiting task is immediately switched in, and the next instruction fetched is for the new task.

Each resident task has a full register set within the register file. The register file is logi-

cally partitioned. No task can write into another task's register file except through a special set of instructions. These instructions have been provided to allow low-overhead intertask communication mechanisms. This requires a large register file. Typical RISC processors have 32 or more general-purpose registers. Thus, an 8-task version of SPArTAn would require 256 registers. In conjunction with the use of IBI and a waiting task model, the use of multiple register sets allows for zero-delay task switching.

In order to detect the impending pipeline dependency, a compile-time dependency detection is program is used. This tool, called ICARUS, reads in the compiler generated assembly language and performs a low level pipeline dependency analysis. ICARUS reads in a database of information about the instruction set. For each instruction type is given the number and position of each input and output operand and the number of clocks required for this instruction to complete. This information is then used to determine when an output operand will not be valid in time to be used as an input operand later in the instruction stream. When this occurs, ICARUS places a tag on the instruction immediately before the instruction which needs the input operand. This tag indicates exactly how many clocks must elapse before the next instruction can be fetched and executed. ICARUS places a tag of 0 on all instructions not involved in a pipeline dependency. Essentially, this technique allows for compile-time detection of all possible pipeline dependencies.

The only restriction is that ICARUS must know exactly how many clocks every instruction requires in order to produce a valid result and write it back to a register. In other words, all instructions must have a completely deterministic delay. This implies that SPArTAn cannot have a cache memory or virtual memory. This is because it is impossible to predict whether a memory reference will be a cache hit or a page fault. It would have to assume the worst case, which is a page fault. Rather than make this poor assumption, SPArTAn has a flat memory system.

In order to get around this problem, a Harvard memory architecture has been used. Instruction memory is separate from Data memory. As noted earlier, scientific applications tend to require much less instruction memory than data memory. This instruction memory is stored in static RAM, and data is stored in 4-way interleaved DRAM. There are a number of benefits to this design decision. First, by separating the instruction memory from the data memory it allows the instruction width to be wider than the data words. This means the instructions can be more horizontal and less decode logic is required in the control unit. Furthermore, the use of more expensive static RAM for the instruction memory means that an instruction-per-clock bandwidth is easily attainable.

The data memory is interleaved to minimize the average latency of the DRAM. Scientific applications tend to have very sequential accesses to memory because of the use of arrays and vectors. This lends itself very well to interleaved memory.

Another important requirement for compile-time dependency detection is that SPArTAn be a pure load/store architecture. The use of memory operands would make it possible to alias operand locations which in turn would make it impossible for ICARUS to identify all pipeline dependencies.

ICARUS also considers all flow-control instructions (conditional branches, jumps, subroutine calls etc.) to be implied pipeline dependencies. Thus, ICARUS automatically

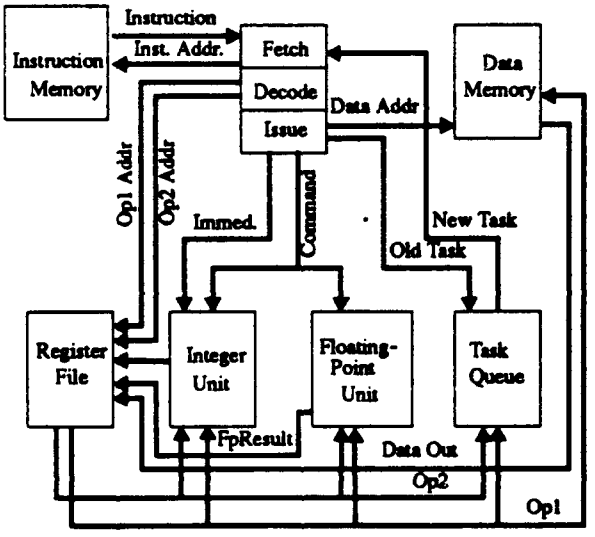
places a non-zero tag on all flow-control instructions.

During execution, the instruction fetch stage reads the tag value on each instruction. If it detects a non-zero tag it begins execution of the waiting task. The task that was running finishes execution of the instruction with the non-zero tag. After this task leaves the control unit it is stored in a queue for continued execution later. The tag value of the instruction which caused the task switch is loaded into a counter in the queue and is decremented. When this value reaches zero the task can continue execution with no pipeline dependencies.

The frequency at which tasks encounter a dependency and switch out is a function of the compiler used and the depth of the pipeline. Study of this factor shows that on average instruction streams encounter a pipeline dependency every 3 to 4 instructions for a pipeline with a maximum depth of 8 stages. This indicates that between 4 and 5 tasks will be required to fill a pipeline on average. In the worst case, every task will encounter a pipeline dependency on every instruction. This implies that $N$ tasks will be needed to fill an $N$-stage pipeline. This is exactly the definition and behavior of the DART processor. So, in a worst case scenario, SPArTAn performs exactly like DART.

Figure 2.1 below shows a block diagram of a particular implementation of SPArTAn. There are 7 major units in SPArTAn: Control Unit, Task Queue, Instruction Memory, Data Memory, Register File, Floating-Point Unit, and Integer Unit.

**FIGURE 2.1.** High-level Block Diagram of SPArTAn




SPArTAn uses a decoupled execution model for its functional units. More precisely, the Data Memory, FPU, and IU are all decoupled from the control unit. The control unit issues commands to a desired execution unit, providing input operands and a return register address for the output. Each of the decoupled units is pipelined (the interleaved data memory is considered pipelined as well). By decoupling the units, it allows for

much easier switching of tasks. The control unit does not have to monitor the progress of each instruction through the entire pipeline.

Each decoupled unit has a separate write port back into the register file. This feature enables the units to have different length pipelines. In many architectures it is necessary for the floating-point and integer pipelines to have the same length in order to avoid register-file conflicts. Having separate write ports for results alleviates this problem. Thus, the integer unit which is more critical to the overall performance of the processor can be done in a two-stage pipeline. The floating-point pipeline has five stages to deal with the more complex floating-point operations.

In addition to the three write ports, two read ports are required to read out dual operands for the floating-point and integer operations. The need for a 5-ported register file is the most difficult portion of the SPArTAn architecture. This register file has been designed and fabricated through MOSIS. The test chip exhibited 25ns read and write access times on all ports. Detailed SPICE analysis indicates that on-chip access times for a 256 word register file is approximately 14ns in a 2.0 micron CMOS technology. Thus, the most difficult part of the SPArTAn architecture is shown to be easily implementable.

### 2.3.3 Software Environment

A complete software environment has been designed and implemented for SPArTan. This environment includes a C compiler, ICARUS, an assembler and linker. In addition, a front-end user interface, called ULYSSES, has been developed to simplify the programming flow for scientific programmers. This interface can manage the creation and compilation of a complete multi-tasking scientific application. ULYSSES also creates custom intertask communication (ITC) routines to allow the tasks to communicate through registers. These mechanisms have zero overhead and provide the capability of modeling any communication topology from square mesh to N-body type systems. ULYSSES operates the compiler, ICARUS, assembler, and linker to produce UNIX compatible, executable programs which can be loaded and run on SPArTAn.

Further research in this area will include a graphical interface to allow scientific programmers to symbolically describe the behavior and interconnection of tasks within an application. This will further simply the work required by the programmer. Work in this are is already underway.

### 2.3.4 Simulation

One of the important byproducts of the SPArTAn research has been the development of SimPLE (Simulator for Processor and Logic Emulation). This simulator is geared towards rapid prototyping of any processor architecture. This is accomplished through the use of flexible macro-blocks which can be instantiated and interconnected to model the behavior of any processor. The user describes the block-level organization of the processor. A special compiler then generates a complete 'C' program to describe the processor model. This model is then compiled using a standard UNIX compiler and is linked with a sophisticated user interface. The user interface allows for single clock operation of the processor for debugging. It also allows for full length simulations of tens of millions of clocks.

SimPLE is able to read in any executable program compiled for the processor and execute the program exactly as it would be on a real processor. This provides the most realistic type of debugging possible. Future work in this area will include more robust graphical interfaces for debugging. In addition, work on an automatic generation of CMOS layout from the SimPLE model is being investigated.

#### 2.3.5 Performance Analysis of SPArTAn

SimPLE was used to model the entire SPArTAn architecture. Programs written using the SPArTAn Software Environment were compiled and run on the model to verify its functionality. Three applications were chosen for performance analysis. All three applications are current research topics in the fields of Physics, Materials Engineering, Electronics and Chemical Engineering. The applications were chosen for their varying use of multi-tasking. This was done to prove the flexible nature of SPArTAn and its value to any scientific application. For more information on these applications consult UC Santa Barbara ECE Technical Report #92-12. In order to demonstrate the improvement in performance provided by SPArTAn, the programs were also run on a SPARC2 workstation (without the use of multi-tasking). A SPARC2 was used for this comparison because it is widely used by scientific researchers.

Comparison was done by measuring the response time of both processors for all three applications.Table 1 below shows the relative performance of both processors to reach the same point in the three applications.

**Table 1: Performance Comparison of SPARC2 vs. SPArTAn (simulated)**

| Processor | Application 1 | Application 2 | Application 3 |
|---|---|---|---|
| SPArTAn | 63 msec | 60 msec | 63 msec |
| SPARC2 | 100 msec | 180 msec | 100 msec |
| Speed-up | 1.57 | 3.00 | 1.57 |

This indicates that on average, SPArTAn is 1.8 times faster than a SPARC2. The response time of SPArTAn was calculated from an example implementation which was designed in full detail. All logic was implemented at the gate-level and critical timing paths were also implemented in CMOS. SPICE analysis indicates that a 30ns clock period is easily attainable on SPArTAn. The SPARC2 was also running at this speed. Note that only user-mode instructions were counted on the SPARC2, excluding all O/S overhead. This policy significantly favors SPARC in the comparisons. On the SPArTAn side, since the hardware is taking care of inter-task context switching and communications, the operating system overhead is essentially included. Both machine times exclude any I/O times.

The simulations of SPArTAn also provided average CPI values for the three applica-

tions. These are given in Table 2 below.

**Table 2: SPArTAn Average CPI (2 million clock simulation)**

| Application 1 | Application 2 | Application 3 |
|---|---|---|
| 1.056 | 1.049 | 1.056 |

It was not possible to measure the CPI of the SPARC2. However, on average the SPARC2 has a CPI of approximately 2.0 or greater. This demonstrates the significant improvement in efficiency provided by IBI. The pipeline in SPArTAn was 94% efficient. The only inefficiency (6%) occurred because of memory-bank conflicts in the data memory. When this occurs, the control unit is stalled to avoid pipeline dependencies which could not be detected at compile time.

#### 2.3.6 Summary

In conclusion, the use of tightly-coupled software and hardware environments allows for the use of Instruction-Block Interleaving and a decoupled execution path. Both zero delay context switching as well as zero-cost branching are both by products of this technique. IBI, coupled with a multi-tasking support at the hardware level, can provide 1.8 times faster response time at the same clock frequency as a common pipelined RISC processor. In addition, the use of a specialized software environment for scientific programmers can provide an excellent foundation of future scientific research.

The techniques used in SPArTAn are limited to a specific implementation. They are a paradigm through which pipeline usage can be optimized to almost theoretically perfect efficiency.

### 2.4 Instruction Scheduling for Decoupled Processors

Wayne Yamamoto, John P. J. Kelly
Computer Architecture and Test Laboratory
University of California, Santa Barbara

Decoupled processors require significantly different instruction scheduling techniques than conventional processors. Decoupled processors are characterized by fine-grained multithreaded execution on a pipeline consisting of decoupled functional units which make the scheduling of instructions a very challenging task. Besides the scheduling of instructions within a given stream, decoupled processors must schedule instructions from multiple streams to promote the optimal use of the processor resources. Thus, the realization of the full potential of a decoupled processor architecture relies heavily on the combination of both static and dynamic instruction scheduling techniques. While many of the techniques are similar to those of conventional pipelined processors, instruction scheduling on a decoupled pipelined processor involves multiple instructions streams and is primarily concerned with high processor utilization rather than single stream execution time. There are also additional constraints. First, instructions executing within the pipeline must be independent, no data or control dependencies, because dynamic, run-time dependency checking does not exist. Second, since the

decoupled processor is targeted for shared memory multiprocessor environments where remote memory access times are non-deterministic, the execution time of instructions is not known a priori.

There are two major elements of decoupled processor instruction scheduling that differentiate it from scheduling on a conventional processor: 1) stream switch policy and 2) stream selection policy. The stream switch policy is used to determine when to context switch and start execution of another stream. Most fine-grained multithreaded systems use a single instruction issue policy where a context switch occurs after every instruction. However, policies which issue a larger block of instructions can reduce the single stream execution time as well as the number of streams required to keep the processor fully utilized. The stream selection policy determines the next stream to execute when a context switch occurs. In a decoupled processor, active streams reside in one of 2 queues: ready or waiting. Only streams in the ready queue can be chosen by the stream selection policy to start execution. A stream is put in the waiting queue if issuing an instruction from it will result in a dependence with another instruction currently executing,. While stream switching and selection are usually carried out in the operating system software, decoupled processors also use the compiler and processor hardware features.

### 2.4.1 FIFO Single Instruction Issue Policy

This is the conventional fine-grained multithreaded scheduling policy. In this policy, only one instruction from a given stream can be executing within the pipeline at a time. The processor issues an instruction from different streams on consecutive cycles. This insures that executing instructions will have no dependencies and therefore hardware dependency checking is not necessary. Selection of the next stream to issue an instruction from is done in a FIFO manner, thus each stream has only a fraction of the processors throughput equal to the reciprocal of the number of active streams. Single stream execution time increases as the number of executing streams increases because the processor throughput is shared equally among the processes. Simulation results of this strategy show high processor utilization in excess of 99% can be achieved provided there is a large enough workload, characterized by the number and types of streams executing. The number of streams required to efficiently use the processor resources is based upon the type of streams, rate at which the processor can issue instructions, and the length of the pipeline. Simulation results show that a workload of 6-10 streams was needed to obtain processor utilization in excess of 99% and the single stream execution time increased from 3-7 times when compared to a conventional processor with similar functional units.

### 2.4.2 Priority Selection Policy

In most fine-grained multithreaded processors, the process of selecting the next stream to execute is determined in a FIFO manner. This was the method chosen by the DART. In many other machines the timing dictates which stream will be executed next. However, in a decoupled pipelined processor, many instruction streams may be eligible to start executing at given time. The choice of the stream selection policy can enhance performance by reducing the single stream execution. The Priority Selection Policy assigns each stream a priority value and selects the stream with the highest priority to execute next from the ready queue. Simulation results show that a 15 to 40% decrease in single

stream execution time can be achieved when compared to the FIFO selection policy.

### 2.4.3 Static Block Scheduling

Static Block Scheduling is a stream switch policy which removes the restriction on allowing only one instruction per stream in the pipeline at any given time without allowing any dependencies between executing instructions. This is accomplished by organizing the instructions within a given stream into fixed-sized blocks of independent instructions. Static Block Scheduling exploits the instruction level parallelism within a stream and allows a block of independent instructions to be consecutively issued from a single stream. After the block has been issued, the processor context switches to the next ready stream and puts the previous stream on the waiting queue until all the instructions from its block have completed execution. This enables each stream to have a larger portion of the processor throughput thereby reducing its single stream execution time. However, since the size of the block is fixed, if there are not enough independent instructions to form a block, no-op instructions are added to pad the block. Early simulation results show that a static block size of 2 gives the highest processor utilization while decreasing the single stream execution time an average of 30%.

### 2.4.4 Tagged Block Scheduling

Within a given application, blocks of independent instructions vary in size. This property is exploited in Tagged Block Scheduling to build variable sized, independent blocks of instructions without using no-ops. Tagged Block Scheduling uses a small tag field that is added to each instruction at compile time to mark the beginning and end of an independent block of instructions. The tag field can be quickly decoded in the instruction fetch stage of the pipeline, thus, does not have to wait for the instruction decode stage. The processor will continue to issue instructions from the same stream until it reads a tag marking the end of a block. At this point, the processor puts the current stream on the waiting queue and switches context to the next ready stream. Streams on the waiting queue are suspended until all instructions from the previous block have completed and then re-enter the ready queue. This policy maximally uses instruction level parallelism within each stream to promote a higher processor utilization than in Static Block Scheduling and a significant decrease in single stream execution time when compared to the single instruction issue policies. Compiler statistics show that the size of the blocks of independent instructions can vary from 1 up to 10 or more and average from 2 to 6. Therefore, Tagged Block Scheduling can provide much better results when compared to Static Block Scheduling.

### 2.4.5 Preemptive Tagged Block Scheduling

This policy combines the Priority Selection Policy with Tagged Block Scheduling to speed up higher priority process while maintaining a high processor efficiency. In Preemptive Tagged Block Scheduling, when a stream with a higher priority than the executing stream enters the ready queue it can preempt the executing stream. Allowing a higher priority stream the ability to preempt a lower priority stream reduces the time it spends in the ready queue and receives a larger fraction of the processor throughput. Thus, the single stream latency is reduced.

### 2.4.6 Simulation Environment

The simulation environment consists of compilation, static instruction scheduling programs, and various simulation programs for performance evaluation. The compilation programs consist of a C-compiler, an assembler, a linker, and a C library. The static instruction scheduling routines run within the compilation programs and perform the Static and Tagged Block Scheduling. There are two different types of simulation programs: an instruction set simulator and the architectural simulators. The instruction set simulator simulates object code produced by the compiler and produces both instruction traces and profiling statistics on the code. The instruction set simulator has an interactive mode which can be used to debug both the software written to benchmark the architecture and the compiler driven instruction scheduling policies. This simulator can execute in excess of 100,000 instruction per second on a Sun-4 and has been used to simulate benchmark programs of substantial size. The architectural simulators simulates the hardware configuration that makes up the processor. The architectural simulator consists of a library of functional units that can easily be configured to model the processor hardware. Instruction traces produced by the instruction set simulator are used by the architectural simulators to obtain performance statistics on the various benchmarks. Synthetic workloads based upon benchmark statistics can also be used to evaluate performance on the various hardware configurations. The simulator runs about 2500 instructions per second and produces performance statistics, including processor utilization and stream execution times.

## 3.0 Circuits Group — Projects and Results

The circuits effort was directed toward one main task and two support tasks. The main task was to investigate dynamic logic for application to VLSI GaAs ICs. The support tasks were to improve SPICE simulation capability (GaAs FET model accuracy) and the speed of full analog transient simulation itself.

When the program began, little work had been reported on dynamic GaAs logic circuits other than some high power, high speed frequency divider circuits [1] which were clearly not suitable for VLSI and some early work on domino logic [2] at UCSB supported under a DARPA contract with JPL. *The circuits study reported here was very successful.* It led to the discovery of an extremely low power, high speed, high density dynamic logic family for GaAs which is called Two-Phase Dynamic FET Logic or TDFL [3][4][5][6]. Using two non-overlapping clocks to eliminate static power dissipation, TDFL offers the potential for low power GaAs VLSI. TDFL gates dissipate only 10 $\mu$W from their 1 V supply when operating at 500 MHz. When the power overhead of the clock generator and driver circuitry is added to that dissipated from $V_{DD}$, the power dissipation figure is still only 35 $\mu$W at 500 MHz. This corresponds to 70 nW/MHz and compares very favorably to both static 5V CMOS (5 $\mu$W/MHz), BICMOS (8 $\mu$W/MHz) and to GaAs DCFL whose gates typically dissipate 200 $\mu$W or more (depending on loading and desired speed). All the standard logic functions (NOT, NAND, NOR, XNOR, AOI) have been demonstrated with TDFL. Also, TDFL is self latching, lending itself to highly efficient pipelined architectures, and is implemented with a standard enhancement/depletion (E/D)-mode MESFET foundry process. Finally, TDFL is

directly compatible with static Direct Coupled FET Logic (DCFL) and superbuffer drivers making its introduction into high speed systems very straightforward. A description of the operation of a TDFL gate and further discussion of its attributes relevant for GaAs VLSI is presented in Sect. 3.1.

In support of circuit design activities, efforts were also made to improve the SPICE MESFET model through accurate microwave characterization of device large signal capacitances. Wafer probe test structures were designed and provided to MOSIS. A unique parameter extraction method was employed to accurately evaluate the voltage dependences of the gate capacitances. Through this work, it was shown that the widely used Statz or Raytheon model [7], while adequate for microwave MESFETS, does not correctly predict capacitances for 1 $\mu m$ enhancement mode MESFETs. Data has been provided to Vitesse, ISI, and Meta-Software. More discussion of this work is found in Sect. 3.2.

Efforts have also been made to improve the simulation capability of SPICE itself. An innovative forward integration method was developed and demonstrated to converge much more rapidly for large networks than the usual Newton or Gear algorithms. A description of this approach is presented in Sect. 3.3.

## 3.1 Two-Phase Dynamic FET Logic

Kevin R. Nary, Stephen I. Long
High-Speed Integrated Circuits Laboratory
University of California, Santa Barbara

### 3.1.1 Gate Operation

The schematic of two TDFL inverters is shown in Fig. 3.1(a) and their operation depicted graphically in Fig. 3.1(b). The gates operate from a single 1.0 V (or greater) power supply and two non-overlapping clocks which toggle between -1.2 V and 0.0 V. In the simulations of Figs. 3.1(b) and 3.2, enhancement mode transistors with a target threshold voltage of 0.05 V and depletion mode transistors with a target threshold voltage of -1.1 V were used. The rise and fall times of the clocks were both 200 ps in these simulations.

When $\Phi_1$ is high, transistors Q1 and Q4 conduct, and the output is charged up to about 1.1 V ( clock high level minus $V_{TD}$) while node A is charged to 0.6 V if the input is high or discharged to 0.0 V if the input is low. After $\Phi_1$ has gone low, $\Phi_2$ goes high and Q3 and Q5 (the input pass transistor to the second inverter) conduct. If node A had been charged high during the $\Phi_1$ phase of operation, then the E-mode FET Q2 conducts, and the output is discharged to ground through Q2 and Q3. (Node C of the second inverter is discharged as well). Had node A been discharged during the $\Phi_1$ high phase, then Q2 remains off, and the output remains high. Note, however, that the output high level settles at about 0.6 V. This is due primarily to gate-to-source conduction of Q6, though charge sharing between the output node and nodes B and C may also play a part.

Note that TDFL gates are sequential gates. That is, the output of a gate is the logical inverse of the input one half clock cycle after the input has arrived. Charge is stored at node A in Fig. 3.1(a) for half of each clock period. While this has the disadvantage that logic signals can only propagate through one level of gates every half clock period, (hence, the effective propagation delay is T/2), this characteristic can be used to considerable advantage. Notably, TDFL gates are particularly suited to pipelined architectures.

Shift registers are very easy to construct and occupy little area since they are just cascades of inverters or other logic elements.

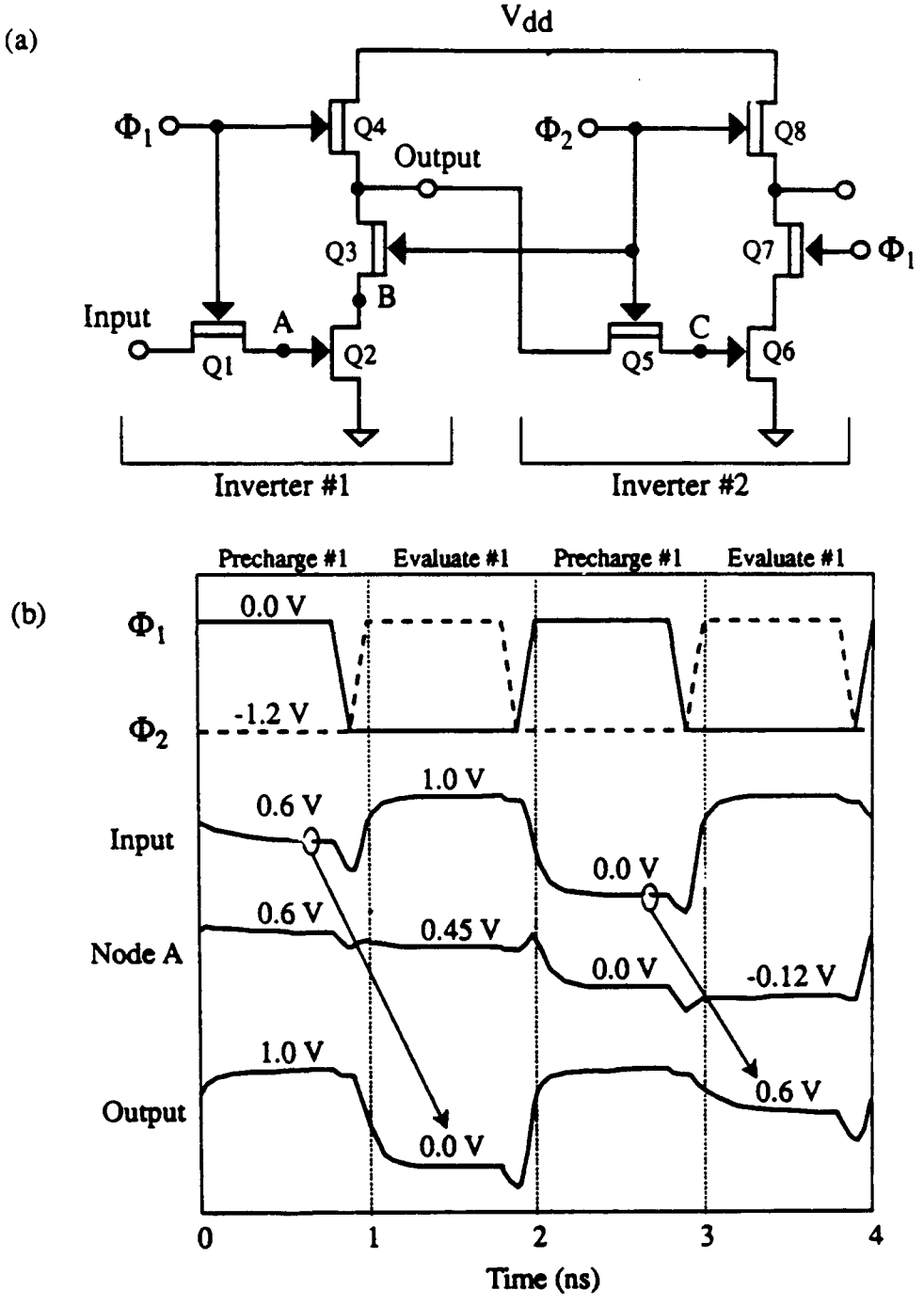



**FIGURE 3.1.** (a) Schematic of two TDFL inverters in series. (b) Simulated operation of inverter with Vdd = 1.0V.

Also, note from Fig. 3.1(b) that TDFL logic levels are compatible with DCFL and super buffer FET logic (SBFL) levels. In fact, the output of a TDFL gate can drive the input of a DCFL or SBFL gate through a pass transistor, and the output of a DCFL or SBFL gate can be directly connected to the input of a TDFL gate. Since the $V_{OL}$ of SBFL gates is lower than that of DCFL gates, their use with TDFL gates is preferable. All three gate types can be operated from the same power supply. This compatibility can be used to great advantage in chip design.

Finally, note that TDFL gates are non-ratioed: logic levels are not determined by the ratio of inverting FET to load FET widths and lengths. This fact affords compact gate layouts through the use of minimum width devices whenever possible.

### 3.1.2 Power Dissipation

Because TDFL is fully dynamic, current flows only during clock transitions. Therefore, power dissipation will be proportional to frequency *f*:

$$P_D = CV^2 f, \qquad \text{(EQ 1)}$$

where *C* is the total node capacitance being switched and *V* is the logic voltage swing.

Figure 3.2 shows the simulated power dissipation versus time of a TDFL inverter operating at 500 MHz. Both the power supplied by $V_{DD}$ and that supplied by the clocks are included in this curve, and the average power dissipated is only 22 $\mu$W when $V_{DD}$ = 1 V and the clock swing ($V_{\Phi PP}$) is 1.2 V. In this figure, the power supplied by the clocks is given by $P_{d\Phi} = |I_\Phi| \times V_{\Phi PP}$. The fan-out in this simulation is 2, and a parasitic layout capacitance of 10 fF has been added to the output. At no point during operation is there a current path from $V_{DD}$ to ground. In terms of power/MHz-gate, this power corresponds to only 44 nW/MHz-gate. TDFL inverters designed to drive 0.1 pF loads dissipate 88 nW/MHz.

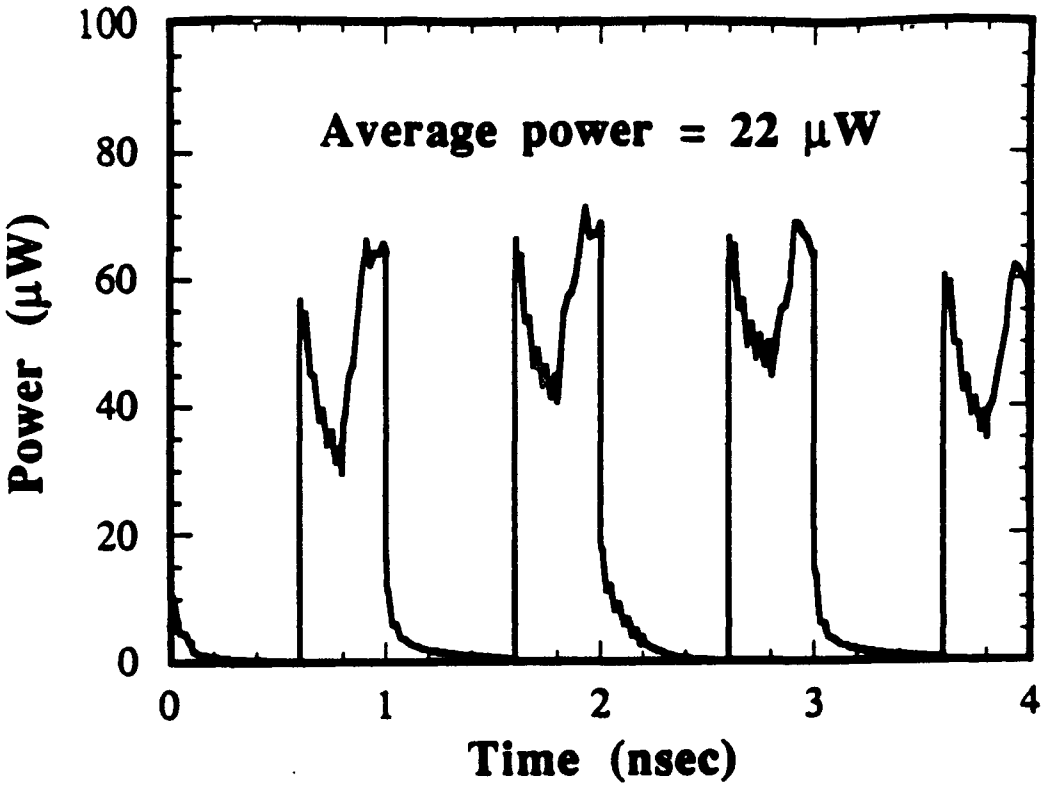



**FIGURE 3.2.** Simulated power dissipation of inverter operating at 500 MHz (fanout = 2, Cload = 10 fF).

For comparison with a representative silicon VLSI technology of the same gate length and similar loading, it has been shown that static 0.8 μm, 5V CMOS requires approximately 5 μW/MHz-gate for 0.1 pF of load capacitance [8]. BiCMOS is slightly higher at about 8 μW/MHz-gate for the same loading and technology. Because CMOS and BiCMOS are static logic approaches, not every gate will be switching on every clock cycle as is the case for TDFL (i.e.; not every CMOS or BiCMOS gate is clocked). The duty factor will vary greatly with the application, but a range of 0.1 to 0.2 may be typical. Even making allowance for the lower duty factor, we see that TDFL power dissipation is *at least 10 times less* than 5 volt CMOS and 15 times less than BiCMOS at any frequency. At the same time, the higher switching speed of GaAs MESFETs over Si MOSFETs with the same gate length will permit maximum clock frequencies at least two times higher than CMOS. Compared with other static low-powr GaAs circuits, TDFL uses roughly 10 times less power than VLSI GaAs DCFL at the 500 MHz frequency simulated.

### 3.1.3 Gate Design

Figures 3.3 and 3.4 show the schematics of TDFL NAND and NOR gates. And-Or-Invert (AOI) gates have also been designed and demonstrated in TDFL [6]. The dimensions, including power, ground and clock busses are 24 μm by 39.2 μm corresponding to a gate density of about 100K gates/cm$^2$. Unlike DCFL, NAND gates are reliable in TDFL because there is never static current flow in the input transistors (Q3 and Q4 in Fig. 3). Proper NAND gate operation is difficult to ensure in DCFL. When the inputs are both high so that static current flows through the input transistors, the drain to source potential of the lower of the two input transistors makes the gate to source potential of the upper transistor significantly less than the gate to source potential of the lower. With a TDFL NAND gate, however, this is not a concern. All that is required for proper operation is the dynamic discharge of the output load capacitance through the series resistance of the input FETs. TDFL NAND gates with three inputs have also been demonstrated.

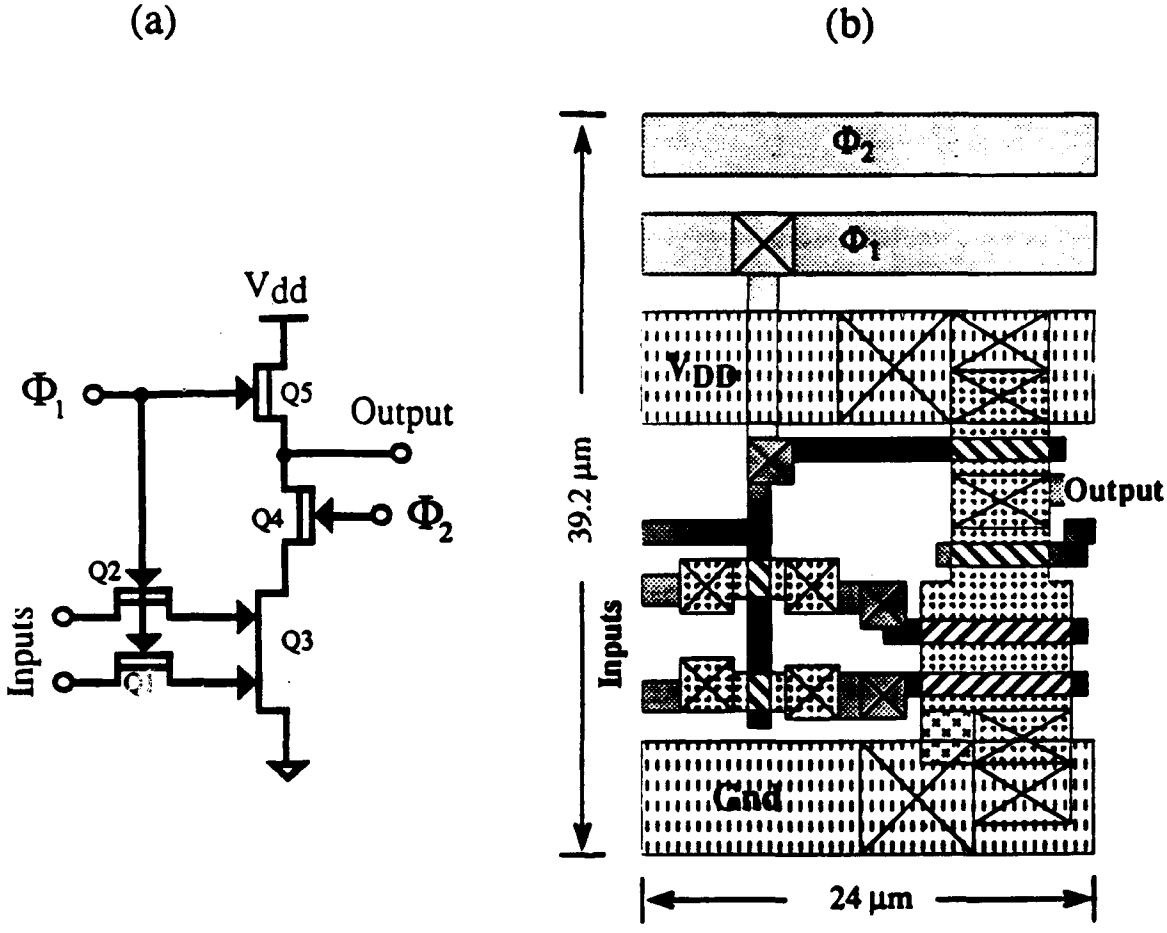



**FIGURE 3.3.** Schematic and layout of two-input TDFL NAND gate.

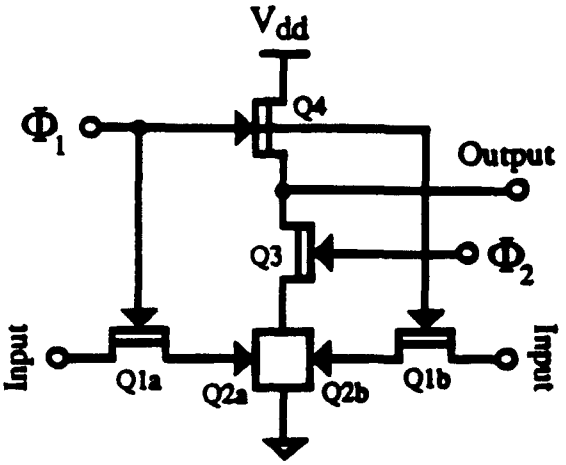



**FIGURE 3.4.** Schematic of two-input TDFL NOR gate.

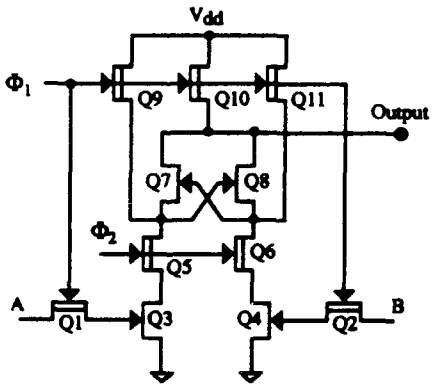



**FIGURE 3.5.** Schematic of fully dynamic XNOR gate which produces an output in only one-half clock cycle.

The schematic of a simple, compact XNOR gate is shown in Fig. 3.5. This XNOR topology is a modification of the circuit found in [9], and it occupies half the area of a DCFL or TDFL XNOR topology made from four NOR gates. Its dimensions are 27.2 $\mu$m by 72.8 $\mu$m (twice the area of a TDFL NAND gate). The key to this XNOR gate's operation is the cross coupled E-mode FETs Q7 and Q8. During the $\Phi_1$ phase, the gates, sources and drains of Q7 and Q8 are pre-charged high. During the $\Phi_2$ phase, the gate to source potentials of both Q7 and Q8 will remain at zero if the inputs are equal, so both FETs are off and the output remains high. If the inputs are not equal, then either Q7 or

Q8 will conduct, and the output will be discharged. Simulation of the XNOR gate with transistor sizes as indicated in Fig. 3.5 predict that the gate will function correctly up to 800 MHz.

### 3.1.4 Design and Performance of TDFL Circuits

Several test circuits have been used to evaluate TDFL performance. These have included the following: chains of gates, linear feedback shift registers, adders, a variable modulus prescaler, and an 8:1 multiplexer. All of these circuits operated with very low power and at clock frequencies in the 500 MHz to 1 GHz range. The adder will be discussed in this report as a representative example of TDFL. For more information regarding the other test circuits, refer to the contract semi-annual reports, to [3], [4], [5], and [6].

The XNOR was evaluated by its use in a four bit ripple carry adder. The 4- bit adder was chosen as a representative MSI demonstration circuit because is easily scalable and there have been several other circuit implementations of such circuits reported in the literature, so the adder provides a good benchmark for comparison between various approaches. The adder is fully pipelined by necessity since TDFL gates are sequential. Two full adder designs based on the XNOR gate of Fig. 3.5 were implemented. Functional block diagrams of these are shown in Fig. 3.6. The adder of Fig. 3.6(a) uses two inverters on the C input and between two of the NAND gates to ensure proper timing. Since signals must propagate through four TDFL gates in the adder of Fig. 3.6(a), its latency is two clock cycles (half a clock cycle per gate). The adder design of Fig. 3.6(b) reduces the latency to one clock cycle by using a TDFL inverter in series with a static SBFL inverter (labeled with an S). These two inverters are used so that the carry input is presented to the input of the sum generating XNOR gate coincidentally with the XNOR of inputs A and B.

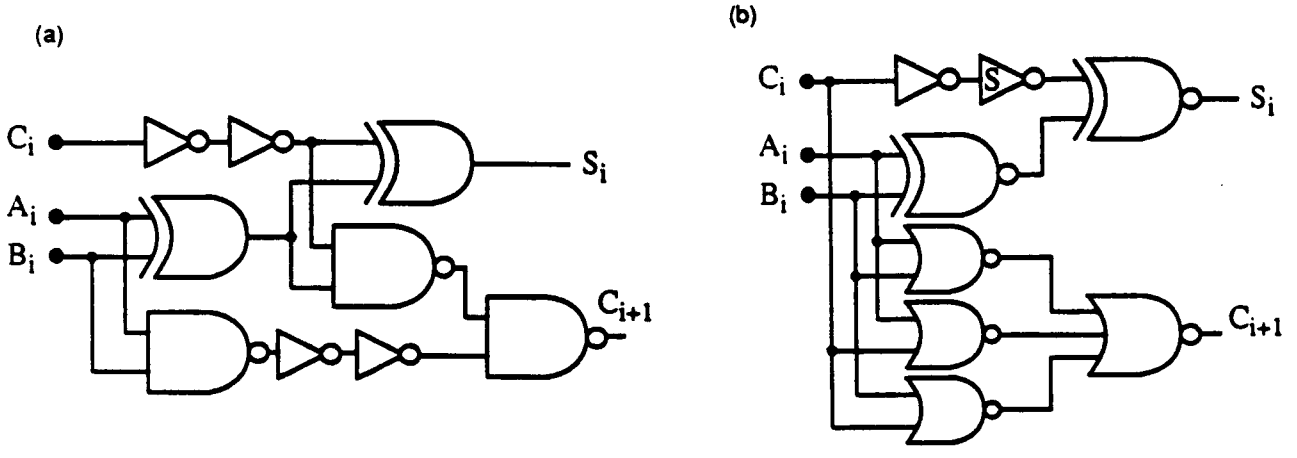



**FIGURE 3.6.** Functional block diagrams of TDFL full adders which produce the sum and carry outputs in two clock cycles (a) and in one clock cycle (b).

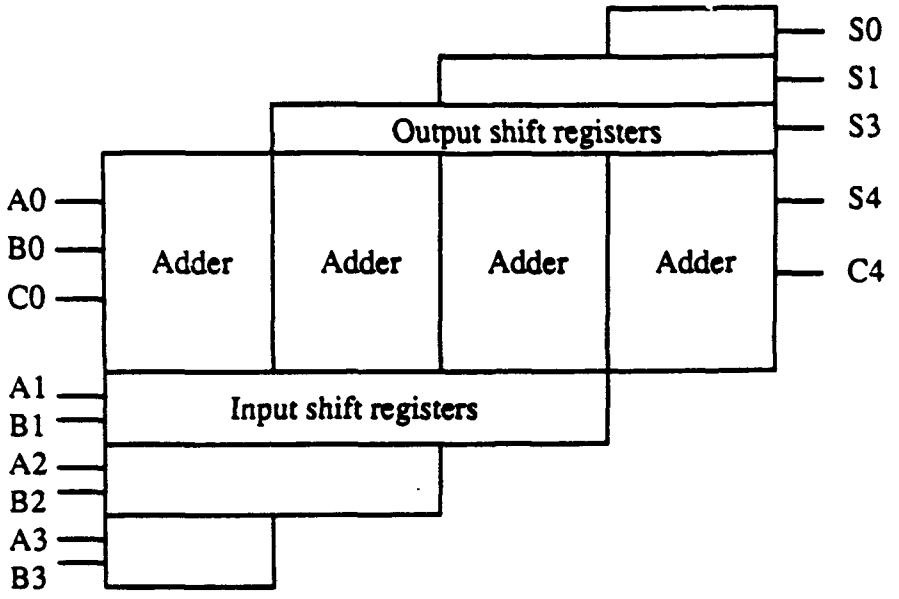



**FIGURE 3.7.** Block diagram of TDFL 4-bit ripple carry adder. Shift registers synchronize data into and out of the adder.

Figure 3.7 depicts a block diagram of the 4-bit ripple-carry adders. Shift registers are used on the inputs and outputs so that all five bits of a sum are shifted out on the same clock cycle. These shift registers are implemented simply by cascading TDFL inverters. Though the latency of these adders may be undesirable in some applications, (i.e. in applications requiring a very fast adder which is used infrequently), they are well suited for highly pipelined, serial applications in which they are used continuously. For example, they are well suited to high speed digital signal processing. The lower latency adder would be important for applications such as direct digital synthesis which can generate an output frequency at half of the clock frequency if the carry is propagated between adders on each clock cycle.

The adders were designed for fabrication in the Vitesse enhancement/depletion 0.8 $\mu$m (HGaAs2) process and were fabricated through the MOSIS/ISI foundry service. The full adder (Type I) of Fig. 3.6(a) is composed of 116 TDFL gates and occupies 0.23 $mm^2$. Its predicted power dissipation at 500 MHz is only 1.3 mW. The 4-bit adder utilizing the design of Fig. 3.6(b) (Type II) uses 64 TDFL gates and 4 SBFL inverters, occupies only 0.16$mm^2$ and has a predicted power dissipation of only 0.8 mW at 500 MHz despite the fact that it uses 4 static gates. These static gates can be designed to be very low power since their propagation delay need only be a little less than half of the intended maximum clock period (i.e.; less than 500 ps). The outputs of the adders are fed into output

buffers through pass transistors which eliminate the precharge phase from the output signals.

Testing of the adders was performed on packaged parts (28 pin leaded chip carriers). A simple 4-layer test board was designed and fabricated which can be rapidly configured to accommodate any pin-out. Fifty ohm transmission lines run to every pin, but jumpers can be installed under the package on the backside of the board if ground or power connections are required. Solder pads for chip capacitors and resistors are also available as needed. Over 25 of these boards have been provided at cost to other users in the university GaAs IC design community.

The two phase clocks were generated off chip using a standard wideband (100 MHz to 3 GHz) 180 degree power hybrid and bias tees. Clock signals were offset below ground so that their crossover point was at or near 50% of the 1.0 to 1.5 volt peak-to-peak excursion. The clock high level could be varied from 0 to 0.3 volts for testing. Higher speed performance is always obtained with more positive clock high logic levels. Vdd was usually 1.0 V, but higher supply voltages can be used with only slight effect on speed. Input signals for testing the adder were generated from the clock by frequency dividers.

The highest demonstrated frequency of operation of the Type I adder was 740 MHz. At this frequency, the power dissipation of the adder was 2.4 mW using a 1.3 V supply. Seven of the ten devices tested operated at 500 MHz, and the average power dissipation at that frequency was 1.08 mW. Testing at higher frequencies was inhibited by the ability to synchronize the adder inputs since the test inputs and the clock signals were provided off-chip. The performance of the five type II adders tested was significantly better. All five operated at 600 MHz, one at 750 MHz, and one at 770 MHz. Slightly higher Vdd potential was needed at the highest frequencies of operation. Fig. 3.9(b) depicts the operation of the type II adder at 770 MHz. a timing diagram of the inputs used to produce the outputs in Fig. 3.8(b) is provided in Fig. 3.8(a). In the oscilloscope photo, the signals are attenuated by 20 dB.

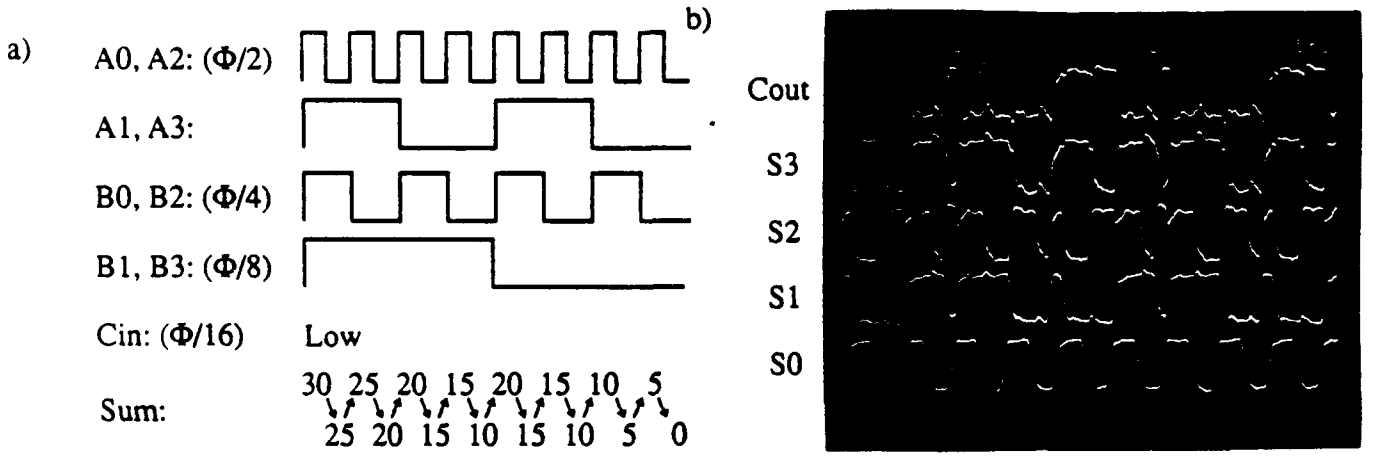



**FIGURE 3.8.** (a) Timing diagram of inputs and expected sum and (b) oscilloscope photo showing outputs from type II TDFL 4-bit adder operating at 770 MHz.

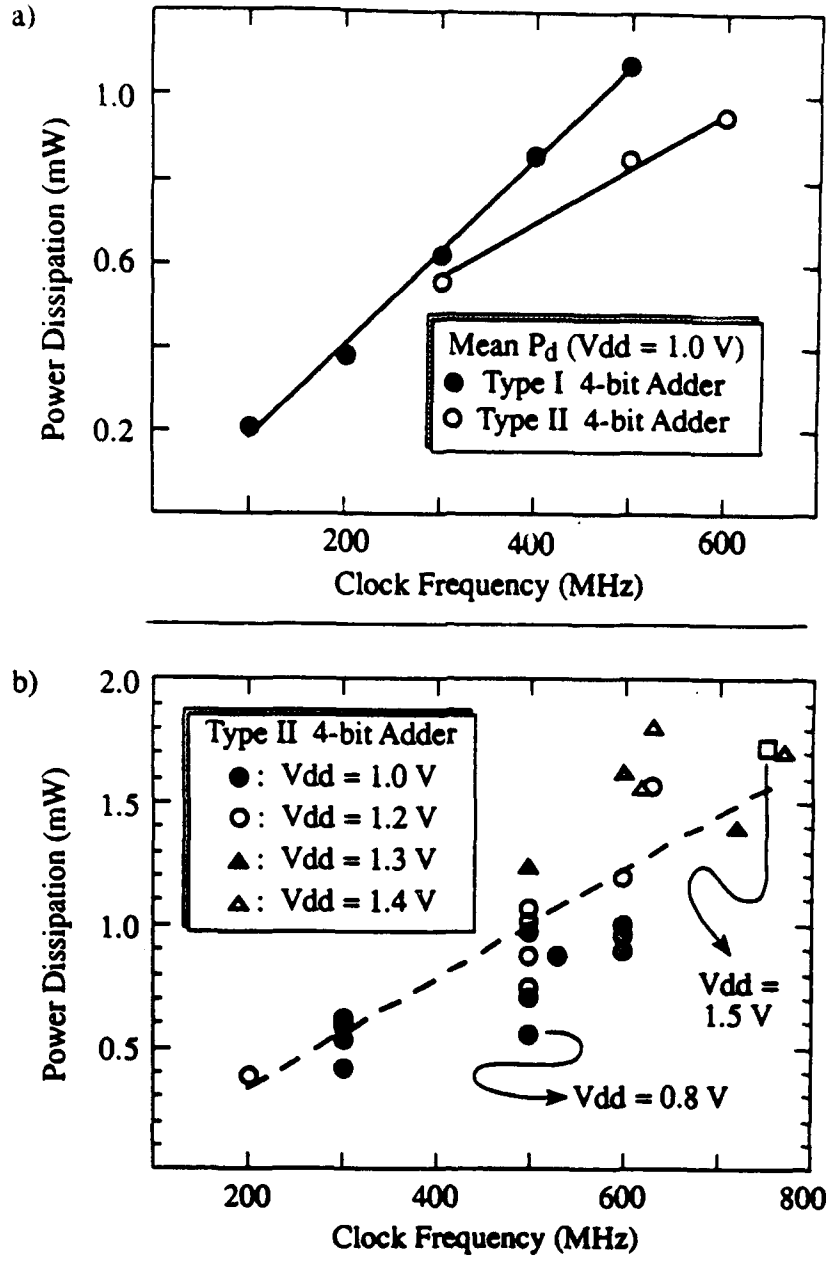



**FIGURE 3.9.** (a) Mean power dissipation versus frequency for type I and type II TDFL adders with Vdd of 1.0V, and (b) scatter plot of power dissipation versus frequency of type II adder.

The average power dissipation versus frequency for the two adder types with Vdd = 1.0V is shown in Fig. 3.9(a). At 500 MHz, the average power dissipated from Vdd (excluding output buffers) was 1.08 mW and 850 mW for the type I and type II adders respectively. The type I adder which operated at 740 MHz dissipated 2.4 mW, while the two type II adders which operated at 750 MHz dissipated 1.7 mW. As expected, the power dissipation is linear with frequency. The line fitted to the type I data has a slope of 2.2 μW/MHz and has an extrapolated zero frequency power dissipation near 0 mW. The fit to the type II data has a slope of 1.3 μW/MHz and a zero frequency power dissipation of 170 μW (the static power consumed by the 4 SBFL inverters used in this design). The power per MHz per gate is 19 nW/MHz/gate and 20 nW/MHz/gate for the type I and II adders respectively. Fig. 3.9(b) is a scatter plot of the power dissipation versus frequency for the type II adder. Included are data points for which Vdd was other than 1.0V.

Table 3 compares the results of this work to 4-bit adders designed in other GaAs logic families. In comparing the TDFL adder results to the others listed in Table 3, recall that the TDFL adders are fully pipelined, whereas the other adders listed have at most one stage of pipelining. If adders made with the other logic families had more stages of pipelining, they would likely have higher maximum frequencies of operation. They would also dissipate more power, however, and would be correspondingly larger. The maximum frequency of operation of the non-TDFL adders listed in Table 1 were obtained from their measured or predicted critical path delay. CCDL (Capacitively Coupled Domino Logic) [10] and TTDL (Trickle Transistor Dynamic Logic) [11] are also dynamic logic families. Both, however, dissipate static power because of the use of static inverters and level shifting circuits. Two other points of comparison, a 1-bit DCFL full adder in the Vitesse Semiconductor cell library dissipates 4.8 mW and has a critical delay (A/B to sum) of 836 ps [12] while a 1-bit full adder in the VTI 1.0 μm CMOS process dissipates 58 μW/MHz and has a critical delay (carry to sum) of 1.59 ns [13].

**Table 3: Comparison of performance of 4 bit adder circuits implemented with different circuit families**

| Logic Family | Area($mm^2$) | Gate Count | Power Diss (mW) | Max Clock Freq (MHz) |
|---|---|---|---|---|
| TDFL type I | | | | |
| TDFL type II | | | | |
| DCFL [10] | 0.32 | 62 | 47 | 714 |
| BFL [10] | 0.75 | 62 | 190 | 500 |
| CCDL [10] | 0.70 | 28 | 96 | 900 |
| TTDL [11] | 0.55 | 15 | 130 | 1250 |
| CMOS [13] | N/A | N/A | 39 | 170 |

To make a fair comparison of the power dissipation of TDFL to other logic families, the power dissipated by the clock drivers needed to operate the adder should be included. The capacitive load presented by the adder to each clock is approximately 600 fF. A two phase clock generator and driver circuit has been designed which dissipates 52 mW when driving 10 pF at 500 MHz. Using a scaled down version of this circuit, and additional 3.1 mW would be added to the 1.1 mW dissipated from Vdd. In terms of power/MHz/gate, this works out to be only 72 nW/MHz/gate. For comparison with a representative silicon VLSI technology of the same gate length and under similar loading, static 5V CMOS gates dissipate approximately 1 μW/MHz (assuming 20 fF of load on each gate) [8].

## 3.2 SPICE MESFET Model

Stephen Long, Peter S. Lassen
High-Speed Integrated Circuits Laboratory
University of California, Santa Barbara

The objective in this study is to accurately measure and extract the voltage dependences of gate capacitances on Vitesse GaAs E/D MESFETs. This large signal capacitance data is important for modeling of analog circuits and ultra-low power dynamic logic circuits. The Statz model [7] presently in use in most versions of SPICE can match the DC measured MESFET data with reasonable accuracy, but the capacitance-voltage dynamic data is not well predicted by this model. By making this data available to MetaSoftware, Vitesse, and Information Sciences Institute we hope to encourage the improvement of MESFET models supported for the MOSIS user's community.

Traditionally, small-signal models of MESFETs are determined from measurements of device S-parameters versus frequency at several bias points, followed by extraction of a small-signal model at each point. However, this method is slow and laborious, often taking days to extract a complete small-signal model. Recently, a new approach for obtaining the bias dependent small-signal device models of GaAs MESFETs has been proposed [14]. Instead of measuring S-parameters versus frequency at a fixed bias point, S-parameters are measured versus bias at a fixed frequency point. This bias scan approach makes GaAs MESFET modelling both more accurate and much faster. The bias scan technique has been implemented on an HP computer, interfacing to the measurement set-up (HP 8510 ANA), and bias dependent device models can be obtained in real-time. No optimization is necessary since the intrinsic FET model has 8 circuit elements and there are 8 measurement variables (the 4 S-parameters) available. A unique solution is always obtained.

As with any microwave measurement technique, accuracy depends on properly treating the parasitics. Here, on-wafer microwave probes are used with the 40 GHz network analyzer, and special test structures were designed (Fig. 3.10) to enable precise extraction of parasitics which allows the intrinsic MESFET equivalent circuit model to be de-embedded from the measured data. The data presented below in Figs. 3.11 and 3.12 have had the probe pad capacitances, inductances, gate, source, and drain resistances removed. The intrinsic layout parasitics (fringing, geometric capacitances) associated with the FET itself are still included in the data.

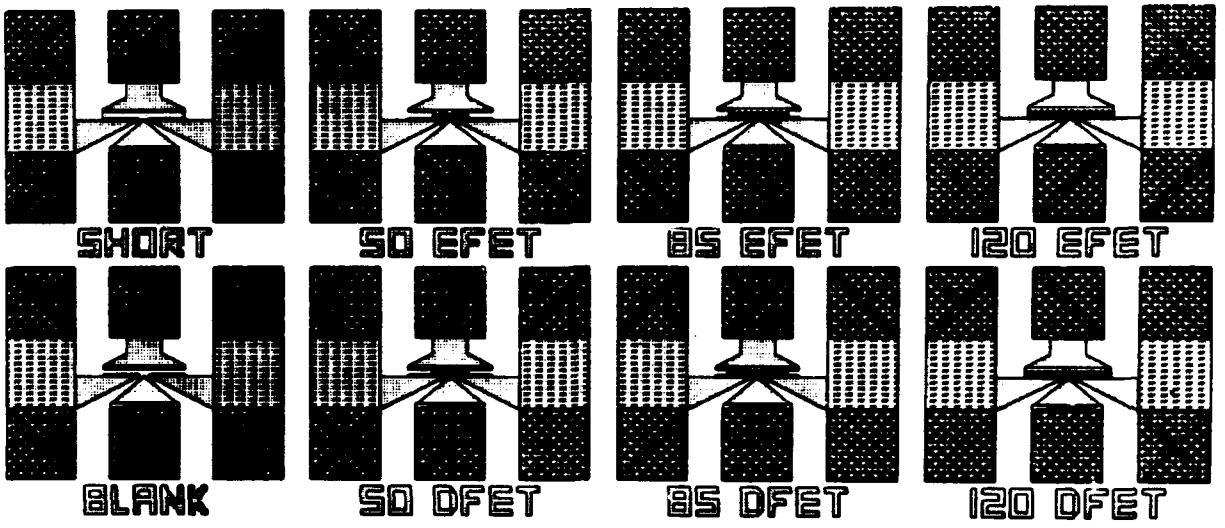



**FIGURE 3.10.** Microwave test structures for evaluation of MESFET model parameters.

Test devices were obtained from two MOSIS GaAs IC runs with Vitesse. The first (M97M) was an early HGaAs2 run from 1989. The second run, from 1992, (also HGaAs2) is labeled N1CD. The measured capacitance data from both runs is attached. The measurements were made by Peter Lassen, a research visitor to UCSB from the Technical University of Denmark, Electromagnetics Institute.

No serious difficulties were encountered in the M97M measurements. The data appeared to be well behaved after extraction with no discontinuities or unexplained features. This data was provided to Meta-software, Vitesse, and ISI in 1992. The voltage dependence of the EFET and DFET gate capacitances are shown in Fig. 3.11.

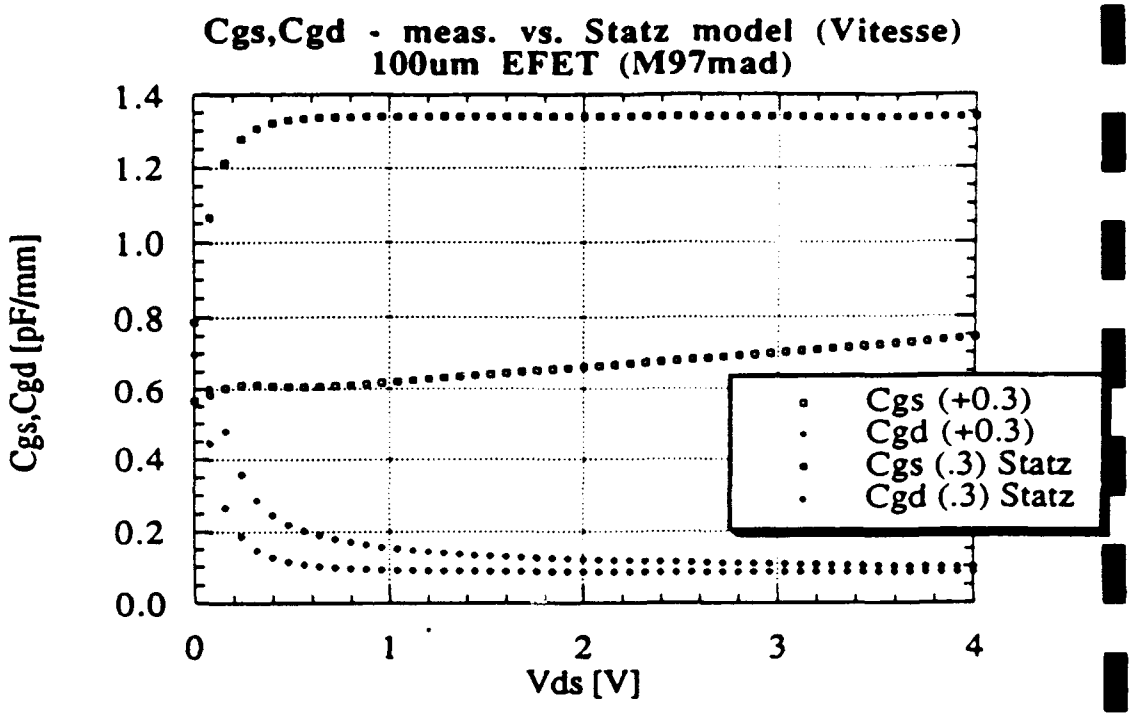



**FIGURE 3.11.** Cgs and Cgd versus Vds with Vgs as a parameter for an enhancement mode MESFET (MOSIS GaAs IC run M97M)

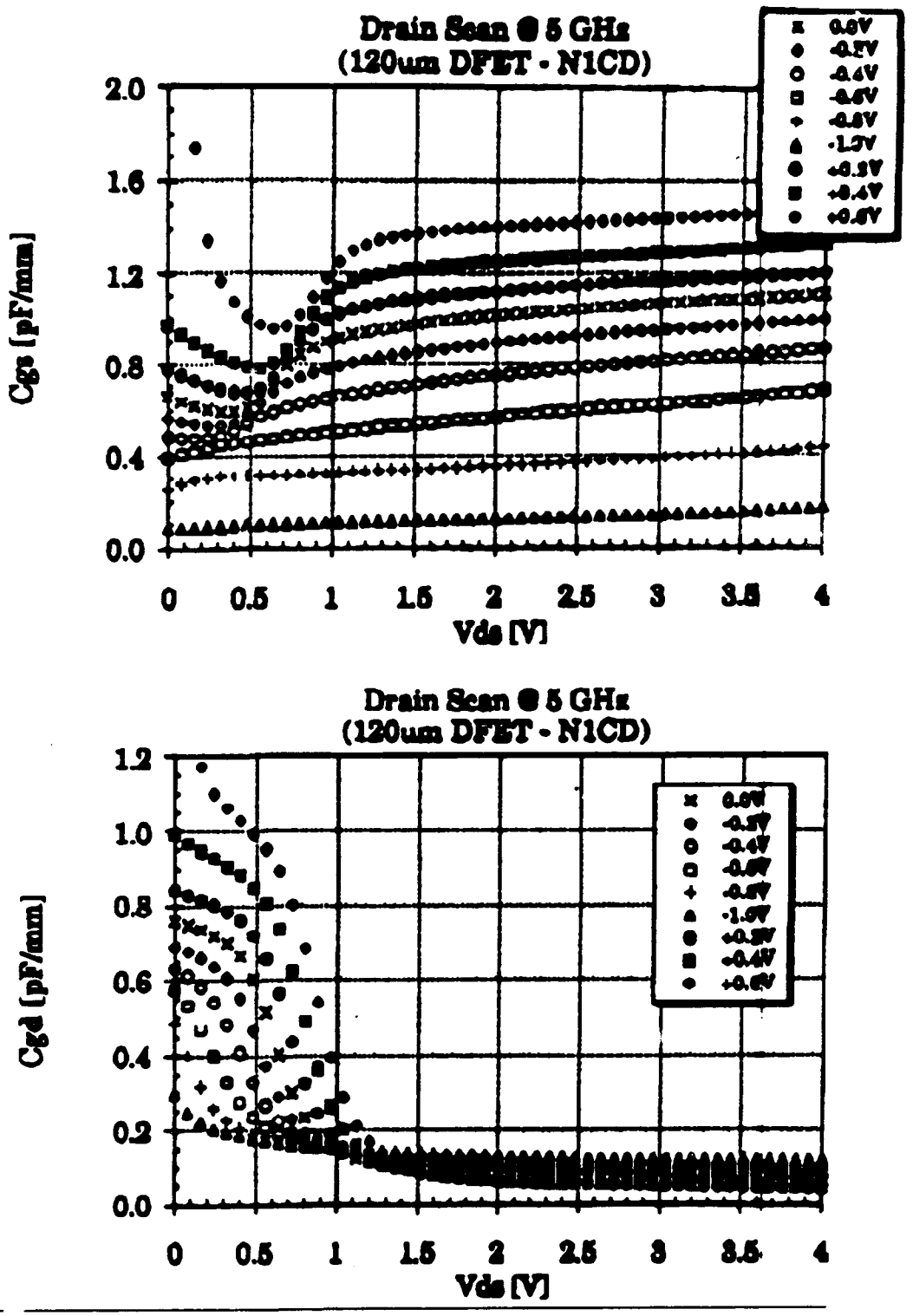



**FIGURE 3.12.** Cgs and Cgd versus Vds with Vgs as a parameter for a depletion mode MESFET (MOSIS GaAs IC run N1CD)

The N1CD measurement data was not as clean. There were divergences in the Cgs data which we initially attributed to breakdown phenomena but later determined that it must be associated with measurement or calibration error. Cgs becomes discontinuous at low Vgs; the curves diverge to infinity. Also, the slope of Cgs in saturation is negative, contrary to previous measurements on M97M and subsequent measurements at 5 GHz on N1CD. All Cgs data measured on N1CD at 10 GHz seem to be affected in this way. The Cgd data appears valid. A subsequent measurement at 5 GHz showed no sign of this problem. Unfortunately, there has not been time available to repeat the complete set of measurements at this frequency. The drain scan for a 120 μm DFET measured at 5 GHz is presented in Fig. 3.12. .

## 3.3 Linear System Decoupling: A Scalable, Stable, Forward Integrating Algorithm for Transient Analysis of LSI Circuit Differential Equations

Robert K. Lewis
Computer Architecture and Test Laboratory
University of California, Santa Barbara

This report describes, in cursory outline, the derivation and essential features of the integration algorithms which lie at the heart of the circuit simulator developed during this project at the Computer Architecture and Test Laboratory in the Department of Electrical and Computer Engineering at the University of California, Santa Barbara under the sponsorship of the Defense Advanced Research Projects Agency. One of our primary results is a new class of integration algorithms for ordinary and partial differential equations which possess the desirable features of being A-stable, decoupled and forward integrating. The fundamental techniques used to attain these goals are the formation of a local linear (i. e. first order) approximation to the circuit equations and a decoupling strategy which allows the resulting linear systems to be iteratively solved as a collection of systems of degree one. Because of these qualities, the set of procedures and ensuing simulation system have been dubbed Linear System Decoupling, LSD. We proceed with a description and derivation of the algorithms.

The fundamental iterative scheme LSD uses to build the solutions of linear systems is now briefly explained. A more detailed treatment may be found in the author's dissertation and a forthcoming paper on this subject. It is well known that the circuit equations may be cast in quasilinear form as

$$C\dot{v} = I \qquad \text{(EQ 1)}$$

with the initial condition

$$v(t_0) = v_0 \qquad \text{(EQ 2)}$$

where $C$ is a symmetric, positive definite, strictly diagonally dominant matrix expressing capacitive couplings and $I$ is the $n$ dimensional vector of node currents; both $C$ and $I$ may depend on the vector of voltages $v$ and the time $t$. The linearization of (1) which is most suitable for the present purposes may be given as

$$C\dot{w} = \nabla_w I - (\nabla_w C) C^{-1} I + \left(\frac{\partial I}{\partial t} - \frac{\partial C}{\partial t} C^{-1} I\right)(t - t_0) + I, \qquad \text{(EQ 3)}$$

$$w(t_0) = 0 \qquad \text{(EQ 4)}$$

In equation (3), it is understood that all occurrences of $I$, $C$ and their derivatives are evaluated at $t = t_0$ and $v = v_0$. For the purposes of this report, it will be helpful to introduce some new notation for some of the terms present in (3). First of all, it will be observed that (3) is a linear ODE of the form

$$C\dot{w} = Pw + a(t - t_0) + c \qquad \text{(EQ 5)}$$

where $P$ is the operator defined by

$$Pw = \nabla_w I - (\nabla_w C) C^{-1} I \qquad \text{(EQ 6)}$$

and $a$ and $c$ are constant real $n$-vectors given by

$$a = \frac{\partial I}{\partial t} - \frac{\partial C}{\partial t} C^{-1} I \qquad \text{(EQ 7)}$$

and

$$c = I. \qquad \text{(EQ 8)}$$

The goal is to establish a technique for constructing solutions to a linear system such as (5) which possesses certain computational efficiencies — in particular, which may avoid the necessity of repeated LU decomposition which is characteristic of the direct method of ODE solution employed by SPICE. While many mathematical and computational approaches to calculating solutions to (5) are known, we reiterate that what is essential here is that if possible the method adopted will *scale* with the number of nodes or variables in the system under discussion. Therefore if possible a technique must be found which allows the solution to (5) to be calculated by successively solving certain *one* dimensional systems which arise from the individual nodes of the circuit at hand. To this end a strategy for *decoupling* the linear system (5) must be developed. Now it may be observed that in this case the matrices $C$ and $P$ are both diagonal (5) decomposes into a set of independent linear equations of the form

$$C_{jj}\dot{w}_j = P_{jj} w_j + a_j(t - t_0) + c_j \qquad \text{(EQ 9)}$$

where the vector components are denoted by the ordinary Roman type face. Of course it is understood that postulating the diagonality of $C$ and $P$ implies

$$C_{ij} = 0 \qquad \text{(EQ 10)}$$

and

$$P_{ij} = 0 \qquad \text{(EQ 11)}$$

for all index values $i$ and $j$ such that $i \neq j$. Equations (9) through (11) provide motivation

for studying the system (5) from the following point of view: a parameter $\varepsilon$ is introduced into the equation (5) which in some sense serves to measure the degree to which this linear ODE may be considered decoupled as is (9). A natural way to accomplish this end, and the one which shall be exploited here, is to permit the off-diagonal entries of the matrices $C$ and $P$ which characterize (5) to depend linearly upon $\varepsilon$. Thus we introduce new matrices $A$, $B$, $C_0$ and $C_1$ with the following properties: $A$ and $B$ are the diagonal and off diagonal parts of $P$, with respective roles being played with regard to the operator $C$ by $C_0$ and $C_1$. These definitions may be formalized in the following equations:

$$A_{ij} = P_{ij}\delta_{ij} \tag{EQ 12}$$

$$C_{0ij} = C_{ij}\delta_{ij} \tag{EQ 13}$$

$$B_{ij} = P_{ij}(1-\delta_{ij}) \tag{EQ 14}$$

$$C_{1ij} = C_{ij}(1-\delta_{ij}) \tag{EQ 15}$$

in which $\delta_{ij}$ is the Kronecker symbol: $\delta_{ij} = 1$ when $i = j$ and is zero otherwise.

Equations (12) through (15) allow the operators $C$ and $P$ occurring in (5) to be expressed as members of smooth (that is to say, infinitely differentiable) *families* of matrices $C(\varepsilon)$ and $P(\varepsilon)$ which are given by

$$C(\varepsilon) = C_0 + \varepsilon C_1 \tag{EQ 16}$$

and

$$P(\varepsilon) = A + \varepsilon B \tag{EQ 17}$$

Then

$$C(0) = C_0, C(1) = C \tag{EQ 18}$$

and

$$P(0) = A, P(1) = P. \tag{EQ 19}$$

For any $\varepsilon$ the linear differential equation (5) becomes

$$C(\varepsilon)\dot{w}(\varepsilon) = P(\varepsilon)w(\varepsilon) + a(t-t_0) + c \tag{EQ 20}$$

and the solution vector $w$ now depends on the variable $\varepsilon$ as well as $t$. If the differential operator $\frac{\partial}{\partial\varepsilon}$ is applied to both sides of this equation, one obtains

$$\begin{aligned}\frac{\partial}{\partial\varepsilon}(C(\varepsilon)\dot{w}(\varepsilon)) &= \frac{\partial}{\partial\varepsilon}(P(\varepsilon)w(\varepsilon) + a(t-t_0) + c) \\ &= \frac{\partial}{\partial\varepsilon}(P(\varepsilon)w(\varepsilon)) + \frac{\partial}{\partial\varepsilon}(a(t-t_0)) + \frac{\partial c}{\partial\varepsilon}\end{aligned} \tag{EQ 21}$$

In this formula the matrix vector multiplications occurring on either side may be

expanded according to the Leibniz rule for derivatives of products, and the terms linear and constant in $t$ may be replaced by zero, since they exhibit no explicit $\varepsilon$ dependence. The result is

$$\frac{\partial C}{\partial \varepsilon}\dot{w} + C\frac{\partial \dot{w}}{\partial \varepsilon} = \frac{\partial P}{\partial \varepsilon}w + P\frac{\partial w}{\partial \varepsilon} \qquad \text{(EQ 22)}$$

We continue with our analysis of the process of taking $\varepsilon$ derivatives begun in (21). Let us introduce a parenthesized superscript notation for $\varepsilon$ derivatives; thus $\frac{\partial^n}{\partial \varepsilon^n} w = w^{(n)}$, and so on. Exploiting this convenience allows (22) to be recast as a linear differential equation for $w^{(1)}(t, \varepsilon)$:

$$C^{(1)}\dot{w} + C\dot{w}^{(1)} = P^{(1)}w + Pw^{(1)}, \qquad \text{(EQ 23)}$$

which is valid for the same set of $t$ and $\varepsilon$ as is (20). Due to the interchangability of the order of the $t$ and $\varepsilon$ derivatives, (23) is in fact a linear ODE for $w^{(1)}$ with $w$ contributing to the driving terms, which may be emphasized by writing it in the form

$$C\dot{w}^{(1)} - Pw^{(1)} = P^{(1)}w - C^{(1)}\dot{w} \qquad \text{(EQ 24)}$$

The initial condition for (23) or (24) is, for any $\varepsilon$,

$$w^{(1)}(t_0, \varepsilon) = 0, \qquad \text{(EQ 25)}$$

which follows by differentiating the initial condition for (20),

$$w(t_0, \varepsilon) = 0, \qquad \text{(EQ 26)}$$

with respect to $\varepsilon$.

The matrix derivatives occurring in (24) may be easily evaluated from the explicit formulae (16) and (17) for $C(\varepsilon)$ and $P(\varepsilon)$. Since these matrix functions are linear in $\varepsilon$ their derivatives are the same for all $\varepsilon$ values; indeed, a simple calculation shows that

$$C^{(1)} = C_1 \qquad \text{(EQ 27)}$$

and

$$P^{(1)} = B, \qquad \text{(EQ 28)}$$

the off-diagonal parts of $C(1) = C$ and $P(1) = P$ respectively. Substituting these results in (24) yields

$$C\dot{w}^{(1)} - Pw^{(1)} = Bw - C_1\dot{w}. \qquad \text{(EQ 29)}$$

Now (29) holds for any value of the parameter $\varepsilon$; that is, the symbols $C$ and $P$ on the left of this equation are to be interpreted as the corresponding functions with the $\varepsilon$ dependence tacitly understood for the sake of notational convenience. In particular (29) applies to the case $\varepsilon = 0$, in which case it becomes

$$C_0 \dot{w}^{(1)} - A w^{(1)} = B w - C_1 \dot{w} \qquad \text{(EQ 30)}$$

Upon examination of this relation the following useful facts may be observed: the operators appearing on the left hand side, $A$ and $C_0$, are diagonal in the coordinate system on v-space which we employ. Furthermore, those occurring on the right, $B$ and $C_1$, have no nonzero diagonal entries in this basis. These considerations conspire together to create a *completely decoupled* linear system of ODEs for the $\varepsilon$ derivative $w^{(1)}$ of $w$ at $\varepsilon = 0$, one in which the component $w_i^{(1)}$ associated with any given node is determined solely by the initial condition $w_i^{(1)}(t_0, 0) = 0$ and the driving functions $w_j(t, 0)$ which are the solutions for the adjacent nodes of the decoupled system (9). Indeed a single component of the vector equation (30) takes the form

$$C_{0ii} \dot{w}_i^{(1)} - A_{ii} w_i^{(1)} = \sum_{j=1}^{N} B_{ij} w_j - \sum_{j=1}^{N} C_{1ij} \dot{w}_j \qquad \text{(EQ 31)}$$

from which it may be seen using the definition of $P$ in terms of circuit elements as given in (6) and the formulas (12) through (15) establishing $A$, $B$, $C_0$, and $C_1$ that a function $w_j(t, 0)$ contributes to the derivative $w_i^{(1)}(t, 0)$ if and only if circuit nodes $i$ and $j$ are connected through some physical device.

We may continue further in the same direction, deriving linear ODEs for higher and higher $\varepsilon$ derivatives of $w$. The starting point for this continuation of our work is equation (29), which as has been pointed out is valid for any value of the parameter $\varepsilon$. The differential operator $\frac{\partial}{\partial \varepsilon}$ may again be applied to both sides of (29), and one obtains

$$\frac{\partial}{\partial \varepsilon}(C \dot{w}^{(1)}) - \frac{\partial}{\partial \varepsilon}(P w^{(1)}) = \frac{\partial}{\partial \varepsilon}(B w) - \frac{\partial}{\partial \varepsilon}(C_1 \dot{w}) \qquad \text{(EQ 32)}$$

The matrices $B$ and $C_1$ on the right hand side of (32) are constant, and thus the derivative may be carried through these operators to yield

$$\frac{\partial}{\partial \varepsilon}(C \dot{w}^{(1)}) - \frac{\partial}{\partial \varepsilon}(P w^{(1)}) = B w^{(1)} - C_1 \dot{w}^{(1)} \qquad \text{(EQ 33)}$$

and now the Leibniz rule may again be employed on the left hand side with the result

$$C^{(1)} \dot{w}^{(1)} + C \dot{w}^{(2)} - P^1 w^{(1)} - P w^{(2)} = B w^{(1)} - C_1 \dot{w}^{(1)} \qquad \text{(EQ 34)}$$

Equation (34) may be further simplified by the introduction of expressions (27) and (28) for the $\varepsilon$ derivatives of $C$ and $P$

$$C_1 \dot{w}^{(1)} + C \dot{w}^{(2)} - B w^{(1)} - P w^{(2)} = B w^{(1)} - C_1 \dot{w}^{(1)} \qquad \text{(EQ 35)}$$

and a final regrouping of terms gives rise to the equation we seek:

$$C_0 \dot{w}^{(2)} - P w^{(2)} = 2 (B w^{(1)} - C_1 \dot{w}^{(1)}), \qquad \text{(EQ 36)}$$

which again holds for any valid value of $\varepsilon$ as does (29). The initial condition placed upon (36) is the analog of (25):

$$w^{(2)}(t_0, \varepsilon) = 0, \qquad \text{(EQ 37)}$$

derived as above by taking $\varepsilon$ derivatives of both sides of that equation. As in the derivation of (30), it is the $\varepsilon = 0$ member of the family of equations (36) which is of central utility for the present purposes, and this is

$$C_0 \dot{w}^{(2)} - A w^{(2)} = 2 (B w^{(1)} - C_1 \dot{w}^{(1)}). \qquad \text{(EQ 38)}$$

In parallel with (30), (38) is also a decoupled linear system of ODEs for $\dot{w}^{(2)}(t, 0)$, and similar remarks concerning its diagonal structure and internode dependence apply. We have thus obtained an easily solvable system for the second functional derivative of the solution family $w(t, \varepsilon)$ in the $\varepsilon$ direction at the value $\varepsilon = 0$. At this point a pattern begins to emerge in the systems of equations for the higher $\varepsilon$ derivatives of the solution functions $w(t, \varepsilon)$. Indeed, careful scrutiny of (29), (30), (36) and (38) leads to the conjecture that the $n$-th $\varepsilon$ derivative of $w(t, \varepsilon)$ satisfies, for general $\varepsilon$,

$$C \dot{w}^{(n)} - P w^{(n)} = n (B w^{(n-1)} - C_1 \dot{w}^{(n-1)}) \qquad \text{(EQ 39)}$$

with the $\varepsilon = 0$ case becoming

$$C_0 \dot{w}^{(n)} - A w^{(n)} = n (B w^{(n-1)} - C_1 \dot{w}^{(n-1)}) \qquad \text{(EQ 40)}$$

A simple inductive argument serves to establish (39) and (40). The cases $n = 1$ and $n = 2$ have been treated individually in the above; assuming (39) holds for *some* $n$, yet another $\varepsilon$ differentiation may be applied. Evaluation of the various derivatives closely parallels the $n = 2$ situation: the constant matrices $B$ and $C_1$ on the right hand side of (39) remain unaffected, and the operator $\frac{\partial}{\partial \varepsilon}$ passes through them to yield

$$\frac{\partial}{\partial \varepsilon} (C \dot{w}^{(n)} - P w^{(n)}) = n (B w^{(n)} - C_1 \dot{w}^{(n)}) \qquad \text{(EQ 41)}$$

The differential-algebraic calculations on the left of (41) may be carried out using exactly the same set of procedures which validated (36): expansion of matrix-vector product derivatives using the Leibniz rule and reliance upon (27) and (28) for evaluation of the matrix derivatives. Indeed one obtains

$$C_1 \dot{w}^{(n)} + C \dot{w}^{(n+1)} - B w^{(n)} - P w^{(n+1)} = n (B w^{(n)} - C_1 \dot{w}^{(n)}) \qquad \text{(EQ 42)}$$

and upon obtaining this result another gathering of identical terms gives rise to the conclusion:

$$C \dot{w}^{(n+1)} - P w^{(n+1)} = (n+1) (B w^{(n)} - C_1 \dot{w}^{(n)}) \qquad \text{(EQ 43)}$$

which is exactly (39) with $n$ replaced by $n+1$. The $\varepsilon = 0$ case of (43) becomes, in a fashion similar to (40), a decoupled system for $w^{(n+1)}(t, 0)$:

$$C_0 \dot{w}^{(n+1)} - A w^{(n+1)} = (n+1) (B w^{(n)} - C_1 \dot{w}^{(n)}) \qquad \text{(EQ 44)}$$

and the proof of (38) and (40) for the general value of $n$ is complete. The initial value of $w^{(n)}$ appropriate to the present development is also easily obtained by taking more $\varepsilon$ derivatives of (26) or (37), and is trivially seen to be

$$w^{(n)}(t_0, \varepsilon) = 0. \qquad \text{(EQ 45)}$$

Equations (9), (44), and (45) encapsulate the essence of our technique for calculating transient solutions of circuit equations. Having built the linearization (3) at any given point $v_0$ and time $t_0$, we first solve the diagonalized version of (3), i. e. (9), which gives $w^{(0)}(t, 0)$. This may be accomplished by any number of known methods, and is generally a simple procedure since explicit formulae for the solution of (9) may be easily presented. Any of several standard approaches may in fact be used to show that $w^{(0)}(t, 0)$ may be written as a linear combination of expressions of the form $e^{\lambda(t-t_0)}p(t)$, where $p(t)$ is a polynomial the coefficients of which, along with $\lambda$, depend on the components of the vector and matrix coefficients occurring in (9). Since the set of such expressions forms a closed algebraic system with respect to the operations involved in solving systems of the generic form (40), all the $w^{(n)}(t, 0)$ may be calculated in a similar fashion. Indeed several related techniques have been developed in the course of this research, but space limitations preclude their inclusion here. All these methods involve direct construction of these solution functions in some form or another, relying on polynomial and exponential evaluation rather than direct numerical integration of (40). It is in this fashion an A-stable, decoupled, forward integrating algorithm for transient analysis of integrated circuits is obtained. The exact solution $w(t, 0)$ of (3) or (5) thus becomes the sum, for $\varepsilon = 1$, of the convergent power series

$$w(t, 0) = \sum_{i=0}^{\infty} \frac{w^{(i)}}{i!}\varepsilon^i \qquad \text{(EQ 46)}$$

where the coefficients are now the functions $w^{(n)}(t, 0)$ of $t$ calculated as outlined above.

A prototype simulator based on the integrator described in this report has been developed. It is currently capable of analyzing circuits containing semiconductor diodes, mosfets, resistors and capacitors. Numerical data concerning its performance can be found in the author's dissertation

## 4.0 Wave Effects Group — Projects and Results

The objective of this activity was to develop CAD tools for analysis of both ideal and lossy on-chip multiconductor transmission lines. As GaAs IC processes have developed, larger numbers of logic gates have been included on a single die. As the die size increases, so does the possiblity that there will be some very long on-chip interconnect lines. These might be for clock distribution or for signal transmission. Losses on these long runs can cause excess delays or logic failures. Long parallel runs can cause crosstalk effects. A means for estimating this undesired coupling and the signal integrity of long lines on-chip is necessary, and the development of a simulation tool to accomplish this was carried out on this project. Design rules were established for the Vitesse GaAs IC process.

While some techniques had been developed at the time this project began for use in the off-chip (PCB) environment, they were either not available to the design community due to proprietary or security restrictions, or were slow and of limited usefulness for on-chip problems, or both. Therefore, UCSB's previous work (performed under another DARPA sponsored project through JPL) which assumed lossless interconnections and linearized driver effective source resistances [15][16] was extended to include calculation of multiconductor line parameters for nonstandard transmission line and dielectric cross sections with losses due to skin effect in thin conductors (quite different than the usual case for thick conductors). Using these line model parameters, it was possible to simulate signal transmission, reflection, and crosstalk effects in high speed IC interconnections using frequency and time domain analysis methods also included in the CAD tool. These tools were to operate efficiently (quickly) and be readily understood by circuit designers who are not necessarily E/M field theory specialists. The simulation tools were also to be verified through measurement of test structures and comparison with slow, but arbitrarily accurate three-dimensional, finite element Poisson solvers.

*All of these goals were accomplished in the program.* Several papers were published describing these results, users documentation was completed, and the CAD tool was distributed to about 15 laboratories free of charge. The features of this program are described in Sect. 4.1. Sect. 4.2 describes the procedure used to determine design rules for the Vitesse process.

### 4.1 Software for the Analysis of High-Speed Digital Circuit Interconnects

Gil Chinn, George Matthaei
High-Speed Integrated Circuits Laboratory
University of California, Santa Barbara

A new CAD tool has been developed on this project for the analysis and design of high-speed lossy multiconductor digital interconnects and similar microwave circuit applications. The programs compute line parameters (L, C, and R matrices) from a physical layout description. These parameters can then be used to simulate the frequency and time domain response to specified input signal(s) on lossy multiconductor lines.

The routines in this program contain a number of novel and very useful features specific

to high-speed GaAs interconnects.

1. Besides treating circuits having one or two common ground planes, our analysis routine can deal with multiconductor lines with coplanar (surface) ground returns. It therefore is capable of analyzing on-chip interconnects as well as circuit boards or MCMs.

2. The capacitance and inductance matrices are computed using an approximate approach developed on this project which is unusually fast and gives excellent accuracy in practical situations. The accuracy has been verified through measurement and comparison with accurate but very slow 3 dimensional Poisson solvers.[17]

3. On-chip interconnect lines are very lossy because of their very small cross-sectional dimensions which are small compared to a skin depth over much of the frequency spectrum of the signals. Our program utilizes a new, highly efficient technique for determining the resistance matrix at high frequencies [18], and adapts an approximation due to Lee and Itoh [19] for modeling the frequency variation of the resistance matrix down to low frequencies. To our knowledge, ours is the only interconnect analysis program that includes this capability.

4. The program includes the internal inductance of the lines (which is a function of the frequency) as a byproduct of the computation of the resistance matrix. It has been shown that if internal inductance is not considered, the rise times calculated for fast pulses may be off by a factor of as much as two [20].

5. The program is modular so that users have considerable flexibility and can utilize the routines separately in other application programs if they wish. Also, it is available for use on either DOS or UNIX platforms.

#### 4.1.1 Comparison with other tools:

Our software is complementary to the EM tools developed at Mayo [21] in that it deals specifically with on-chip interconnections and the unique problems that accompany this case (coplanar geometries, conductors with very small cross sections such that their resistance is very high and varies widely with frequency). The portion of our program which deals with finding the C, L, and R matrices is unusually fast [17][18]. For instance a two-line microstrip configuration analyzed using a commercial software package [22] took 6 min. to get the C and L matrices when run on a PC with an 8086 processor, while our program took only 10 sec. when run on the same computer. Meanwhile the accuracy achieved by the commercial program was 2 to 3 percent while the accuracy of our results was better that 1 percent as compared to results from a finite-element program using thousands of "elements."

### 4.2 Modeling of High-Speed Digital IC Interconnections

K. Belarre, K. Nary, S. Long
High-Speed Integrated Circuits Laboratory
University of California, Santa Barbara

This research was focused on modeling and formulating design rules for long interconnects on GaAs substrates with a buried p-type implant. The objective was to identify

those specific conditions for the Vitesse Semiconductor H-GaAs-2 process under which signal propagation could be obtained without excessive attenuation or delay. While strict time-of-flight propagation was difficult to obtain in all cases of interest, tradeoffs were identified which preserved the signal integrity and maintained acceptable propagation delay on the line. Extensions to the HGaAs3 technology were made. A Master's Thesis was written describing the results of this work [23].

Microwave measurements were made on transmission line test structures to determine the influence of the buried p-type layer on signal propagation [24]. The effects caused by this layer were observed to be dominant at low frequencies, thereby yielding frequency dependent line parameters for the RLGC equivalent circuit often used for a model of lossy transmission lines. A physically intuitive equivalent circuit model called the LSUB model (for lossy substrate) was constructed by adding three more elements to account for the influence of the p- layer. This model was shown to accurately represent the s-parameter data versus frequency measured for the CPW test structures fabricated in the Vitesse H-GaAs-2 process. However, the values of the added elements were not predictable from the measurement or the physical structure, and therefore the model was not scalable with changes in the cross sectional dimensions of the line. Lacking time and sufficient GaAs area for test structure fabrication (each structure was 0.5 x 10 mm), efforts were made to compare the time domain predictions of the RLGC model with the LSUB model to determine whether meaningful propagation delay and attenuation predictions could be obtained using the simpler, but scalable model.

As a result of this comparison, a simple scalable RLGC model was obtained for the metal 2 and the metal 3 lines from a software program which predicts the line parameters of an RLGC model for a given cross sectional geometry of a multiple line configuration (see section 4.1 of this report). This model matches well with the LSUB model in the time domain simulations on HSPICE. However, its applicability to lines of different dimensions will be confirmed only after comparing test simulation results of this model for lines of different dimensions with those of the LSUB models of the corresponding lines. The LSUB models for lines of other cross-sectional dimensions could not be obtained because test structures of other dimensions were unavailable for measurement.

The scalable line models were used to determine optimal line dimensions for signal propagation along long interconnects. The issues of power, delay, area, and their relative trade-offs were considered in the design of interconnects and line drivers. It was concluded that the least expensive design option is to use lines of minimum cross sectional dimensions. For a given line length and load, test simulations were run to study the influence of line drivers with different output resistances on signal propagation. The LSUB model was available for lines of minimum dimensions, and was used in these simulations. Charts that indicate the costs of power dissipation, line delay, and line attenuation as a function of source resistance for given line lengths and loads were prepared. This set of charts gives a clear picture of the trade-offs between delay, power dissipation and available noise margin for a realistic range of line lengths and loads. These design rule guidelines are independent of the logic family of the circuits used.

These design rules, however, specifically apply to lines fabricated in the Vitesse H-GaAs-2 process. Vitesse Semiconductor has now adopted the H-GaAs-3 process. An extra layer of metal (metal 4), which comprises a conducting ground plane extending over the entire chip above the metal 3 layer, is included in the new process. In addition,

minimum linewidths are narrower leading to increased resistance per unit length. Therefore, significant differences in line parameters are expected for the new process. Transmission line test structures in the H-GaAs-3 process were unavailable for measurement, however based on the successful use of a simplified line model with H-GaAs-2, a RLGC model for H-GaAs-3 was calculated using our line parameter program. It was found that the parasitic effect of the p-layer was rendered insignificant by the stronger influence of the metal 4 ground plane. Hence the RLGC models for lines in the H-GaAs-3 process from the line parameter program were assumed to be sufficiently accurate, even though the program could only represent the p-layer as a conducting plane.

Various line configurations were simulated using this model, and design rules for lines fabricated in H-GaAs-3 were predicted. As the metal 4 layer provided a ground return path, no benefit was found in designing the interconnect lines as CPW structures. Striplines of minimum width were found to be least expensive in terms of chip area required for adequate signal propagation. The attenuation and delay were predicted and were found to be generally worse with H-GaAs-3 than with H-GaAs-2 due primarily to the increased series resistance of minimum geometry lines and the reduced line impedances produced by the metal 4 ground plane. Charts similar to those for lines in the H-GaAs-2 process were prepared to predict signal propagation characteristics of striplines of minimum width in the new process.

## 5.0 References and Bibliography

[1] M. Rocchi and B. Gabillard, "GaAs Digital Dynamic IC's for Applications up to 10 GHz," IEEE J. Solid-State Cir., vol. SC-18 (3) pp. 369-376, 1983.

[2] L. Yang, R. Chakharapani and S.I. Long, "A High-Speed Dynamic Domino Circuit Implemented with GaAs MESFET's," IEEE J. Solid-State Cir., vol. SC-22 (5) pp. 874-879, 1987.

[3] K.R. Nary and S.I. Long, "GaAs Two-Phase Dynamic FET Logic: A Low-Power Logic Family for VLSI," IEEE J. Solid-State Cir., vol. 27 (10) pp. 1364-1371, 1992.

[4] K.R. Nary and S.I. Long, "A 1 mW, 500 MHz 4-bit Adder Using Two-Phase Dynamic FET Logic Gates" IEEE GaAs IC Symp. Miami Beach, FL, pp. 97-100, 1992.

[5] K.R. Nary, "Gallium Arsenide Metal-Semiconductor Field Effect Transistor Dynamic Logic Gate Topologies", Ph.D. Dissertation, University of California, Santa Barbara, 1992.

[6] P.S. Lassen, S.I. Long and K.R. Nary, "Ultra-low Power GaAs MESFET MSI Circuits Using Two-Phase Dynamic FET Logic," IEEE J. Solid-State Cir., vol. 28 (10) pp. 1993.

[7] H. Statz, P. Newman, I. Smith, R. Pucel and H. Haus, "GaAs FET Device and Circuit Simulation in SPICE," IEEE Trans. Elect. Dev., vol. ED-34 (2) pp. 160-169, 1987.

[8] J.D. Gallia, "High-Performance BiCMOS 100K-Gate Array," IEEE J. Solid-State

Circuits, vol. SC-25 (2) pp. 142-148, 1990.

[9] L.R. Lau, S.C. Pi and W.L. Stahl, "Inverse Exclusive OR Circuit for Dynamic Logic," IBM Technical Disclosure Bulletin, vol. 17 (6) pp. 1974.

[10] D.H.K. Hoe and C.A.T. Salama, "Dynamic GaAs Capacitively Coupled Domino Logic,," IEEE J. Solid-State Circuits,, vol. SC-26 (6) pp. 844-849, 1991.

[11] D.H.K. Hoe and C.A.T. Salama, "GaAs Trickle Transistor Dynamic Logic," IEEE J. Solid-State Circuits, vol. SC-26 (6) pp. 1441-1448, 1991.

[12] Vitesse Semiconductor, "GaAs DCFL ASIC Design, Appl. Note 7" 1992 Product Data Book. pp. 8 - 36, 1992.

[13] VLSI Technology, Inc., "1.0 micron CMOS VSC370 Portable Library - Rev. 2.0"

[14] B. Hughes and P.J. Tasker, "Bias dependence of the MODFET intrinsic model elements values at microwave frequencies," IEEE Trans. on Elect. Dev., vol. 36 (10) pp. 2267-2273, 1989.

[15] G.L. Matthaei, S.I. Long and C.-H. Shu, "Simplified Linear Representation of Logic Gate Terminal Impedances for Use in Interconnect Crosstalk Calculations," IEEE J. Solid-State Cir., vol. SC-24 (5) pp. 1468-1470, 1989.

[16] G.L. Matthaei, C.-H. Shu and S.I. Long, "Simplified Calculation of Wave Coupling Between Lines in High-Speed Integrated Circuits," IEEE Trans. on Circuits and Systems, vol. 37 (10) pp. 1201-1208, 1990.

[17] G.L. Matthaei, G.C. Chinn, C.H. Plott and N. Dagli, "A Simplified Means for Computation of Interconnect Distributed Capacitances and Inductances," IEEE Trans. on Comp. Aided Design, vol. 11 (4) pp. 513- 524, 1992.

[18] G.L. Matthaei and G.C. Chinn, "Approximate Calculation of the High-Frequency Resistance Matrix for Multiple Coupled Lines" 1992 IEEE Intern. Microwave Symp. Albuquerque, NM, pp. 1992.

[19] H.-Y. Lee and T. Itoh, "Phenomenological Loss Equivalence Method for Planar Quasi-TEM Transmission Lines with a Thin Normal Conductor or Superconductor," IEEE Trans. Microwave Theory and Techniques, vol. 37 (12) pp. 1904-1909, 1989.

[20] T.R. Arabi, "On the Modeling of Conductor and Substrate Losses in Multiconductor, Multidielectric Transmission Line Systems," IEEE Trans. MTT, vol. 39 (7) pp. 1090-1097, 1991.

[21] G.-W. Pan, J.A. Prentice, S.K. Zahn and A.J. Staniszewski, "The simulation of high-speed, high-density digital interconnects in single chip packages and multichip modules," IEEE Transactions on Components, Hybrids, and Manufacturing Technology, vol. 15 (4) pp. 465-77, 1992.

[22] A. Djordjevic, "Matrix Parameters for Multiconductor Transmission Lines: Soft-

ware and User's Manual", Artech House, Norwood, MA, 1989.

[23] K.G. Bellare, Master's Thesis, University of California, Santa Barbara, 1992.

[24] K.R. Nary, K.G. Bellare and S.I. Long, "A Model for Coplanar Waveguide Transmission Line Structures on Semiconductor Substrates," IEEE Trans. on Microwave Theory and Tech., vol. 41 (accepted for publication) 1993.

# 6.0 Program Statistics

## 6.1 Publications from Contract no. N00014-88-K-0497

G.L. Matthaei, S.I. Long, and C.-H. Shu, "Simplified Linear Representation of Logic Gate Terminal Impedances for Use in Interconnect Crosstalk Calculations", IEEE Journal of Solid-State Circuits, vol. 24, pp. 1468-1470, October 1989.

S. I. Long, S. E. Butner, "GaAs IC Fabrication with MOSIS -- A User's Perspective", Proc. IEEE Int. Symp. on Circuits and Systems, New Orleans, May 1990, pp. 2562-2565.

S. I. Long and S. E. Butner, "Gallium Arsenide Digital Integrated Circuit Design" McGraw-Hill Publishing Co., New York, 1990. ISBN 0-07-038687-0

J.P.J. Kelly, T. McVittie, S. Murphy, "Techniques for Building Dependable Distributed Systems: Multi-Version Software Testing", Proc. Fault Tolerant Computing Symposium (20) at Newcastle Upon Tyne, UK, June 1990.

G.L. Matthaei, K. Kiziloglu, N. Dagli, S.I. Long, "The Nature of the Charges, Currents, and Fields in and About Conductors Having Cross-sectional Dimensions of the Order ot a Skin Depth", IEEE Trans. on Microwave Theory and Techniques, vol. MTT-38, pp. 1031-1036, Aug. 1990.

J.M. Dodd, "The Design and Analysis of SPINN: A Scalable, Fault-Tolerant Processor Interconnection Network", MS thesis/technical report, UCSB ECE Department, 1990.

A. Young, "Gallium Arsenide Pass Transistor Integrated Circuits", MS thesis/technical report, UCSB ECE Department, 1990.

G.L. Matthaei, C.-H. Shu, S.I. Long, "Simplified Calculation of Wave Coupling Between Lines in High-Speed Integrated Circuits", IEEE Trans. on Circuits and Systems, vol. 37, pp. 1201-1208, October 1990.

T.I. McVittie, J.P.J. Kelly, W. Yamamoto, "An Empirical Investigation of the Effect of Formal Specifications on Program Diversity", 2nd IFIP Working Conference on Dependable Computing for Critical Applications, Tucson, AZ, Feb. 1991

J.P.J. Kelly, W.I. Yamamoto, "ACCM: Atomic Concurrency Control Memory for a Database Machine", UCSB/ECE Dependable Computing Lab Technical Report, 1991.

J.P.J. Kelly, T.I. McVittie, W.I. Yamamoto, "Implementing Design Diversity to Achieve Fault Tolerance", IEEE Software, 1991

D.J. Fouts, "Theory, Design, and Simulation of GASP: A Block Data Flow Architecture for Gallium Arsenide Super Computers" Ph.D, dissertation/technical report, UCSB ECE Department 1991.

S. Butner, S. Bordelon, L. Endres, J. Dodd, and J. Shetler, "A Fault-Tolerant GaAs/CMOS Interconnection Network for Scalable Multiprocessors", IEEE Journal of Solid-State Circuits, vol. 26, no. 5, May 1991, pp. 692-705

K. Kiziloglu, N. Dagli, G.L. Matthaei, and S.I. Long, "Experimental Determination of High-Speed GaAs Digital Circuit Interconnect Parameters," IEEE MTT-S Digest, OF-II-11, 639-641, June 1991

J. Shetler and S. Butner, "Multiple Stream Execution on the DART Processor", Proceedings of the International Conference on Parallel Processing, Aug 1991, pp. I92-I96.

K. Kiziloglu, N. Dagli, G.L. Matthaei, S.I. Long, "Experimental Analysis of Wave Effects in High-Speed GaAs Digital Circuit Interconnects", IEEE Transactions on Microwave Theory and Techniques, August 1991.

Joy S. Shetler, "DART: A Decoupled Computer Architecture", PhD dissertation, UCSB ECE Dept., August 1991

K. Kiziloglu, N. Dagli, G.L. Matthaei, and S.I. Long, "Experimental Analysis of Transmission Line Parameters in High-Speed GaAs Digital Circuit Interconnects," IEEE Trans. Microwave Theory and Techniques, 39 (8) 1361-1367, Aug 1991

D. Fouts and S. Butner, "Architecture and Design of a 500 MHz Gallium-Arsenide Processing Element for a Parallel Supercomputer", IEEE Journal of Solid-State Circuits, vol. 26, no. 9, Sept 1991, pp. 1199-1211

K.R. Nary and S.I. Long, "Two-Phase Dynamic FET Logic: An Extremely Low Power, High Speed Logic Family for GaAs VLSI," 1991 IEEE GaAs Symposium, pp. 83-86, Monterey, CA October, 1991

K.R. Nary and S.I. Long, "Dynamic Latch for High Speed GaAs Domino Circuits," IEE Electronics Letters, 28 (1) 36-37, January 1992

Y. Wang, A. Mangaser, P. Srinivasan, S. Jordan, S. Butner, "The 3DP: A Processor Architecture for Three-Dimensional Applications", IEEE Computer, vol 25, no. 1, January 1992, pp. 25-36.

K. G. Bellare, "Modeling and Design of Interconnect Lines on Lossy GaAs Substrates," Master of Science Thesis, Univ. of California, Santa Barbara, March 1992. Also available as a technical report from the ECE Department.

D. J. Fouts, "A Gallium-arsenide Digital Phase Shifter for Clock and Control Signal Distribution in High-Speed Digital Systems," IEEE J. Solid State Circuits, SC-27, pp.

802 - 809, May 1992.

G.L. Matthaei and G.C. Chinn, "Appropriate Calculation of the High-Frequency Resistance Matrix for Multiple Coupled Lines," presented at the 1992 IEEE International Microwave Symposium in Albuquerque, NM, June, 1992.

J. Dodd, "Design and Implementation of SPArTAn: A Scientific Processor Architecture and Software Enviroment for Task-Partitioned Applications", PhD dissertation, UCSB ECE Dept., June 1992.

O. Vainio, M. Sundaram, S. Long, and Y. Neuvo, "A 600 MHz Median-Type Digital Filter on GaAs, IEEE J. Solid State Circuits, SC-27, pp. 940 - 943, June 1992.

K. R. Nary, "GaAs Two-Phase Dynamic FET Logic: A Very Low Power Logic Family for VLSI," Ph.D. dissertation, Univ. of California, Santa Barbara, June 1992. Also available as a technical report from the ECE Department.

K.R. Nary and Stephen I. Long, "GaAs Two-Phase Dynamic FET Logic: A Low Power Family for VLSI," IEEE J. Solid State circuits, vol. 27(10), October 1992, pp. 1364-1371.

K.R. Nary and Stephen I. Long, "A 1 mW, 500 MHz 4-Bit Adder Using Two-Phase Dynamic Fet Logic Gates," presented at the 1992 IEEE GaAs IC Symposium, October 1992, pp. 92-100.

K.G. Bellare, K.R. Nary, and S.I. Long, "A Model for Coplanar Transmission Lines on GaAs Substrates with p- Implant Layers," Accepted for publication in IEEE Microwave Theory and Techniques, 1993.

## 6.2 Graduate Students Supported

| | | |
|---|---|---|
| K. Bellare | S. Bordelon | G. Chinn |
| J. Dodd | L. Endres | D. Fouts |
| J. Johnson | S. Jordan | K. Kiziloglu |
| R. Lewis | T. McVittie | K. Nary |
| J. Shetler | W. Yamamoto | |

## 6.3 Visiting Scholars

| | | |
|---|---|---|
| P. Lassen | J. Malek | J. Madsen |
| O. Vainio | | |